Fundamentals of Computing and Artificial Intelligence

Unit-wise Assignment Question Bank

with Sample Answers

Course Code:	CSAI1101
Programme:	B.Tech. CSE (AI & ML) — Semester I
Department:	Computer Science & Engineering
Institution:	Sharda University, Greater Noida
Faculty:	Dr. Gopal Chandra Jana
Coverage:	Units 1 to 5
Question Levels:	2, 4, 6, 8 and 12 marks

Total Questions: 375 (**75 per unit**)
Per unit: 15 questions each at 2, 4, 6, 8 and 12 marks
Marks per unit: 15(2+4+6+8+12) = **480** **Grand total: 2400 marks**

Course page: https://www.gcjana.in/courses/shardauniversity/2601/fundamentals-of-computing-and-artificial-intelligence/

Post your doubts: https://www.gcjana.in/courses/shardauniversity/2601/fundamentals-of-computing-and-artificial-intelligence/#Post-doubts

September 4, 2026

General Instructions to Students

1. All assignments are to be **handwritten** unless stated otherwise, and submitted as a **single PDF** through the course page.
2. For every numerical question, **show all intermediate steps**. A correct final answer without working carries at most 40% of the marks.
3. **State the formula** before substituting values, and carry units where applicable.
4. Where a diagram is asked for, it must be **neatly labelled**; unlabelled diagrams earn no marks.
5. **Assignment 1** covers Units 1 and 2; **Assignment 2** covers Units 3 and 4; **Assignment 3** covers Unit 5 and the presentation.
6. Copied or identical submissions will be awarded **zero** for all parties.
7. Write the **question number and marks** clearly before each answer.
8. The sample answers here are **concise model outlines**. In your submission, expand them to the length suggested below in your own words.

Marks	Expected length	Expected content
2	3–4 lines	A crisp definition, statement or a single-step answer
4	∼half page	Definition/statement + one example or one short calculation
6	∼1 page	Explanation + a small diagram or a two-step numerical
8	∼1–1.5 pages	Full explanation + labelled diagram or complete numerical
12	2–2.5 pages	Multi-part answer: explanation + diagram + numerical + discussion

How to use this bank. Each unit has 75 questions arranged by marks. Questions are mapped to the course outcomes (CO1–CO6) and Bloom's levels (K1–K4) used in the session plan. Attempt a mix of levels from every unit. Numerical answers are rounded sensibly; always show your working.

Contents

Contents

1 Unit 1 — Hardware Aspect of Computer Science & Engineering

Syllabus. **A.** History of computing systems; computer basics and organisation. **B.** Data representation: number systems with conversion, binary codes. **C.** Evolution of programming languages: machine, assembly, low-level and high-level languages. *(CO1–CO6, K1–K3)*

Section A — 2 Mark Questions

(Very short answers; attempt all fifteen)

Q1 **[2 Marks]**

Define a computer. Name the four operations of the information-processing cycle.

Sample Answer. A **computer** is an electronic device that accepts data, processes it under a set of stored instructions, produces information, and can store it. The four operations are **input**, **processing**, **output** and **storage**.

Q2 **[2 Marks]**

Differentiate between data and information with one example.

Sample Answer. **Data** are raw, unprocessed facts (e.g. marks 78, 92, 65); **information** is processed, meaningful data (e.g. average = 78.3). Processing converts data into information.

Q3 **[2 Marks]**

Who is called the "Father of the Computer" and what machine did he design?

Sample Answer. **Charles Babbage**; he designed the **Analytical Engine** (1837), the first design of a general-purpose computer.

Q4 **[2 Marks]**

State the stored-program concept in one line, and name the scientist associated with it.

Sample Answer. The **stored-program concept** keeps the program in memory alongside the data, so the machine runs a new task simply by loading a new program. It is credited to **John von Neumann**.

Q5 **[2 Marks]**

Name the electronic component that defines each of the first four generations of computers.

Sample Answer. 1st — **vacuum tubes**; 2nd — **transistors**; 3rd — **integrated circuits (ICs)**; 4th — **microprocessors**.

Q6 **[2 Marks]**

Convert $(1101)_2$ to decimal.

Sample Answer. $(1101)_2 = 1\times8 + 1\times4 + 0\times2 + 1\times1 = 8 + 4 + 0 + 1 = \mathbf{(13)_{10}}$.

Q7 **[2 Marks]**

Convert $(25)_{10}$ to binary.

Sample Answer. Repeated division by 2 gives remainders $1, 0, 0, 1, 1$ (read upwards) $\Rightarrow (25)_{10} = \mathbf{(11001)_2}$.

Q8 **[2 Marks]**

How many bits are in one byte, and how many bytes in one kilobyte (KB)?

Sample Answer. One byte = **8 bits**; one KB = **1024 bytes** ($= 2^{10}$ bytes).

Q9 **[2 Marks]**

Name the three components of the CPU.

Sample Answer. The **Control Unit (CU)**, the **Arithmetic Logic Unit (ALU)**, and the **Registers**.

Q10 **[2 Marks]**

State one difference between RAM and ROM.

Sample Answer. **RAM** is volatile (contents lost when power is off) and is read/write working memory; **ROM** is non-volatile (retains contents) and mainly stores start-up instructions.

Q11 **[2 Marks]**

Write the ASCII codes (in decimal) of the characters 'A' and 'a'.

Sample Answer. **'A'** = 65 and **'a'** = 97; they differ by 32, the "case" gap.

Q12 **[2 Marks]**

What is the hexadecimal digit for decimal 12, and for decimal 15?

Sample Answer. Decimal 12 = **C**; decimal 15 = **F**.

Q13 **[2 Marks]**

Name the three levels of programming languages, from closest to hardware to closest to humans.

Sample Answer. **Machine language** (binary) → **assembly language** (mnemonics) → **high-level language** (near-English).

Q14 **[2 Marks]**

What is the job of an assembler?

Sample Answer. An **assembler** translates **assembly-language** mnemonics into **machine code** (binary) that the CPU can execute.

Q15 **[2 Marks]**

State the three steps of the machine (fetch–execute) cycle.

Sample Answer. **Fetch** the instruction → **Decode** it → **Execute** it; then the cycle repeats.

Section B — 4 Mark Questions

(Short answers with an example; attempt all fifteen)

Q1 **[4 Marks]**

Explain the Input–Process–Output (IPO) model of a computer with a simple example.

Sample Answer. In the **IPO model**, a computer takes **input** (data), performs **processing** under program control, and gives **output** (information), optionally using **storage**. Example: entering three marks (input), the CPU computes their average (process), and the screen shows 78.3 (output). The IPO cycle underlies every program.

Q2 **[4 Marks]**

List and briefly explain any four characteristics of a computer.

Sample Answer. **Speed** — billions of operations per second; **Accuracy** — correct results for correct input and logic; **Diligence** — no tiredness or drop in performance; **Versatility** — the same machine does many different tasks. (Others: storage, automation.) These make computers powerful general tools, though they still lack intelligence of their own.

Q3 **[4 Marks]**

Describe the two limitations of a computer captured by the phrase "GIGO".

Sample Answer. **GIGO** = *Garbage In, Garbage Out*: a computer has **no intelligence of its own** and only follows instructions, so (i) **wrong input** produces wrong output, and (ii) it **cannot judge or correct** its own logic. Hence the quality of results depends entirely on the quality of the input and the program.

Q4 **[4 Marks]**

Compare the first and second generations of computers on component, size and language.

Sample Answer. **1st generation** used **vacuum tubes**, were huge and hot, and were programmed in **machine language**; e.g. ENIAC. **2nd generation** used **transistors**, were smaller, faster and cooler, and used **assembly** and early high-level languages (FORTRAN, COBOL); e.g.

IBM 1401. The move from tubes to transistors was the first big step in miniaturisation.

Q5 **[4 Marks]**

Convert $(215)_{10}$ into binary, octal and hexadecimal.

Sample Answer. Binary: $(215)_{10} = (11010111)_2$. Group in 3s for octal: $011\,010\,111 = (327)_8$. Group in 4s for hex: $1101\,0111 = (D7)_{16}$. *Check:* $D7 = 13{\times}16 + 7 = 215$. ✓

Q6 **[4 Marks]**

Convert $(2F)_{16}$ to decimal and to binary.

Sample Answer. Decimal: $(2F)_{16} = 2{\times}16 + 15 = 32 + 15 = (47)_{10}$ (F = 15). Binary: write each hex digit as 4 bits: $2 = 0010$, $F = 1111 \Rightarrow (00101111)_2$.

Q7 **[4 Marks]**

Add the binary numbers $(1101)_2$ and $(0110)_2$; verify your answer in decimal.

Sample Answer. Column addition with carries gives $1101{+}0110 = (10011)_2$. Check: $13{+}6 = 19$, and $(10011)_2 = 16 + 2 + 1 = 19$. ✓

Q8 **[4 Marks]**

Explain BCD coding and encode $(59)_{10}$ in BCD.

Sample Answer. **Binary Coded Decimal (BCD)** codes each decimal digit separately in 4 bits. For $(59)_{10}$: $5 = 0101$, $9 = 1001 \Rightarrow (0101\,1001)_{\text{BCD}}$. Note BCD differs from pure binary $(59)_{10} = 111011$, and uses only patterns 0000–1001.

Q9 **[4 Marks]**

Explain the von Neumann architecture with the role of the system bus.

Sample Answer. The **von Neumann architecture** has a single **memory** holding both program and data, a **CPU** (CU + ALU + registers), and **I/O**, all connected by **buses** that carry data, addresses and control signals. Because instructions and data share one path to memory, a bottleneck (the "von Neumann bottleneck") can limit speed.

Q10 **[4 Marks]**

Explain the memory hierarchy and why it is shaped like a pyramid.

Sample Answer. From top to bottom: **registers**, **cache**, **main memory (RAM)**, **secondary storage**. Going up: faster, costlier, smaller; going down: slower, cheaper, larger. The pyramid shape reflects this trade-off — a little very-fast memory near the CPU and a lot of cheap slow storage below.

Q11 **[4 Marks]**

Differentiate machine language and assembly language on four points.

Sample Answer. **Machine language** is pure binary, directly executed, fastest, but unreadable and error-prone. **Assembly language** uses mnemonics (ADD, MOV), is far more readable, but needs an **assembler** to translate it. Both are **machine-dependent** (tied to a CPU family), unlike high-level languages.

Q12 **[4 Marks]**

Differentiate a compiler and an interpreter with one example each.

Sample Answer. A **compiler** translates the *whole* program at once into machine code (e.g. C, C++), giving faster execution. An **interpreter** translates and runs *line by line* (e.g. Python), which is slower but easier to test. A compiler reports errors at the end; an interpreter stops at the first error.

Q13 **[4 Marks]**

State the five generations of programming languages with one example of each.

Sample Answer. **1GL** machine (binary); **2GL** assembly (MOV, ADD); **3GL** high-level, e.g. C/Java/Python; **4GL** very-high-level/problem-oriented, e.g. SQL; **5GL** AI/declarative, e.g. Prolog. Each generation moves closer to human thinking and further from raw binary.

Q14 **[4 Marks]**

State Moore's Law and explain its practical significance.

Sample Answer. **Moore's Law** (1965) observes that the number of transistors on a chip roughly **doubles about every two years**. Practically, this means computers keep getting smaller, faster and cheaper, which has driven the growth of PCs, smartphones and modern AI.

Q15 **[4 Marks]**

Explain the difference between hardware and software, giving two examples of each.

Sample Answer. **Hardware** is the physical, touchable part of a computer (e.g. CPU, RAM, keyboard, monitor). **Software** is the set of programs/instructions (e.g. the operating system, a browser). Hardware is the "body", software the "soul" — neither is useful without the other.

Section C — 6 Mark Questions

(Explanation with a small diagram or a two-step numerical; attempt all fifteen)

Q1 **[6 Marks]**

Explain the five functional units of a computer with a neat block diagram.

Sample Answer. The five units are the **input unit**, **memory unit**, **ALU**, **control unit** and **output unit**. Input feeds data to memory; the CU fetches and directs; the ALU performs arithmetic and logic; results go to memory and then to output. (Diagram: Input → Memory ↔ CPU [CU+ALU] → Output, with the CU sending control signals to all units.) Together, ALU + CU + registers form the CPU, the brain of the computer.

Q2 **[6 Marks]**

Explain the fetch–decode–execute cycle with a diagram and state the role of the clock.

Sample Answer. The CPU repeats: **Fetch** (get the next instruction from memory using the program counter) → **Decode** (the CU interprets it) → **Execute** (the ALU/registers carry it out), then loops back. (Diagram: a 3-box cycle with an arrow back to Fetch.) The **clock** drives this loop; a 3 GHz clock ticks 3 billion times per second, and each tick can trigger work, so a faster clock generally means more instructions per second.

Q3 **[6 Marks]**

Convert $(45.25)_{10}$ to binary, showing the integer and fraction parts separately.

Sample Answer. **Integer 45:** repeated division by 2 gives remainders $1, 0, 1, 1, 0, 1$ (upwards) $\Rightarrow (101101)_2$. **Fraction 0.25:** multiply by 2: $0.25\times2 = 0.5$ (0), $0.5\times2 = 1.0$ (1) $\Rightarrow (.01)_2$. Combining: $(45.25)_{10} = \mathbf{(101101.01)_2}$. *Check:* $32+8+4+1 = 45$ and $0.25 = 2^{-2}$. ✓

Q4 **[6 Marks]**

Explain 1's complement and 2's complement, and represent -6 in 8-bit 2's complement.

Sample Answer. For a negative number we use a **sign bit** (1 = negative). **1's complement** inverts every bit of the positive value; **2's complement** adds 1 to the 1's complement. For -6 (8-bit): $+6 = 00000110$; invert → 11111001; add 1 → **11111010**. Modern computers use 2's complement because subtraction becomes addition and there is a single zero.

Q5 **[6 Marks]**

Perform $13 - 7$ using 5-bit 2's complement arithmetic, showing every step.

Sample Answer. $13 - 7 = 13 + (-7)$. $+7 = 00111$; invert → 11000; add 1 → 11001 $(= -7)$. Now add: $01101\,(13) + 11001\,(-7) = 1\,00110$. Discard the carry out of the 5th bit $\Rightarrow 00110 = \mathbf{6}$, which is $13 - 7$. ✓

Q6 **[6 Marks]**

Explain ASCII and Unicode and state why Unicode was needed. Encode the word "Hi".

Sample Answer. **ASCII** uses 7 bits ($2^7 = 128$ characters) for English letters, digits and symbols; extended ASCII uses 8 bits. It cannot represent Hindi, Tamil, Chinese or emoji, so **Unicode** was created to encode *every* writing system (UTF-8 is the common, ASCII-compatible form). "Hi": H = 72 = 01001000, i = 105 = 01101001.

Q7 **[6 Marks]**

Explain Gray code and write the 3-bit Gray code sequence for 0 to 7. Why is it useful?

Sample Answer. In **Gray code**, successive numbers differ by exactly **one bit**, which avoids read errors in position sensors and encoders. Sequence 0–7: $000, 001, 011, 010, 110, 111, 101, 100$. In ordinary binary, $3 \to 4$ flips three bits at once ($011 \to 100$); Gray code flips only one, so a sensor can never briefly show a wrong value.

Q8 **[6 Marks]**

Classify computers by size, giving one use of each class.

Sample Answer. **Supercomputer** — fastest; weather forecasting, research, AI training (e.g. PARAM). **Mainframe** — large organisations serving thousands of users (banks, railways). **Minicomputer** — mid-sized, for a department. **Microcomputer** — personal computers, laptops, tablets, smartphones. Size, speed and cost decrease down this list while the number of users typically decreases too.

Q9 **[6 Marks]**

Explain how a colour image is stored in a computer, with a small calculation.

Sample Answer. An image is a grid of **pixels**; each pixel's colour is three values — **Red, Green, Blue** — each 0–255, i.e. **8 bits** each. So one pixel = 3×8 = 24 bits = 3 bytes; pure red = $(255, 0, 0)$. A 1000×1000 image needs 1000×1000×3 = 3,000,000 bytes ≈ 3 MB. This shows how all media reduce to binary codes.

Q10 **[6 Marks]**

Explain the working of an assembler with a diagram, and give two reasons assembly is still used.

Sample Answer. An **assembler** converts assembly mnemonics into machine code (Diagram: Assembly source → Assembler → Machine code). Each mnemonic maps to an opcode, e.g. `MOV AL, 61h` → `10110000 01100001`. Assembly is still used for **device drivers/embedded systems** and where **tight speed or direct hardware control** is essential.

Q11 **[6 Marks]**

Show the "add two numbers" operation at all three language levels and comment on the trade-off.

Sample Answer. **High-level:** `c = a + b`. **Assembly:** `MOV AX,a` / `ADD AX,b`. **Machine:** a binary opcode such as `00000011`... As we move down, code gets faster to execute but harder to read and less portable; as we move up, it gets easier and portable but slightly slower (a translator sits in between). High-level languages trade a little speed for a lot of productivity.

Q12 **[6 Marks]**

Summarise the generations of computers in a table (component, feature, example).

Sample Answer.

Gen	Component	Feature	Example
1st	Vacuum tubes	Huge, hot, machine code	ENIAC
2nd	Transistors	Smaller, assembly	IBM 1401
3rd	ICs	OS, time-sharing	IBM System/360
4th	Microprocessors	PCs, GUI, VLSI	Intel 8086, IBM PC
5th	AI / ULSI	Learning, reasoning	Modern AI systems

The clear trend is smaller, faster, cheaper, more reliable and lower power each generation.

Q13 **[6 Marks]**

Explain the units of memory from bit to terabyte, and compute the number of characters in 2 KB.

Sample Answer. **Bit** (0/1) → **nibble** (4 bits) → **byte** (8 bits, one character) → KB → MB → GB → TB, each step $\times 1024$ ($= 2^{10}$). Since one byte stores one character, $2\,\text{KB} = 2 \times 1024 =$ **2048** characters.

Q14 **[6 Marks]**

Explain the roles of system software and application software, with two examples of each and the role of the OS.

Sample Answer. **System software** manages the machine — the operating system, compilers, device drivers (e.g. Windows, Linux). **Application software** performs user tasks — a browser, MS Word, games. The **operating system** is the key system software: it manages memory, files, processes and devices and acts as the bridge between the user and the hardware.

Q15 **[6 Marks]**

A file contains 500 characters of English text stored in extended ASCII. Compute its size in bits and bytes, and explain.

Sample Answer. Extended ASCII uses **8 bits (1 byte) per character**. So 500 characters = 500 bytes = $500 \times 8 =$ **4000 bits**. In KB: $500/1024 \approx 0.49$ KB. This illustrates how text size scales linearly with the number of characters.

Section D — 8 Mark Questions

(Full explanation with diagram or complete numerical; attempt all fifteen)

Q1 **[8 Marks]**

Trace the historical evolution of computing from early tools to the stored-program computer, naming at least six milestones.

Sample Answer. **Abacus** (∼3000 BC) — first counting tool; **Napier's Bones** (1617) and **Pascaline** (1642) — mechanical calculators; **Babbage's Analytical Engine** (1837) — first general-purpose design, with Ada Lovelace as first programmer; **ENIAC** (1945) — first general-purpose electronic computer (∼18,000 vacuum tubes); **von Neumann's stored-program idea** (1945) — program stored in memory; **UNIVAC** (1951) — first commercial computer. The trend runs mechanical → electronic → stored-program, each stage adding speed and flexibility.

Q2 **[8 Marks]**

Explain the von Neumann architecture in detail with a labelled diagram, and state its main limitation.

Sample Answer. The architecture has four parts: **CPU** (CU + ALU + registers), **memory** (holds program *and* data), **I/O**, and **buses**. (Diagram: CPU ↔ Memory ↔ I/O over a shared system bus; inside the CPU, CU and ALU with registers.) The CU fetches and decodes; the ALU computes; registers hold current values. Its main limitation is the **von Neumann bottleneck**: since data and instructions share one memory path, that path limits speed. This is the model behind almost all computers.

Q3 **[8 Marks]**

Explain, with all steps, the conversion of $(347)_{10}$ to binary, octal and hexadecimal, and verify each.

Sample Answer. **Binary:** $347 \div 2$ remainders give $(101011011)_2$. **Octal:** group in 3s: $101\,011\,011 = (533)_8$. **Hex:** group in 4s: $0001\,0101\,1011 = (15B)_{16}$. *Verify:* $(533)_8 = 5{\times}64 + 3{\times}8 + 3 = 320 + 24 + 3 = 347$; $(15B)_{16} = 1{\times}256 + 5{\times}16 + 11 = 256 + 80 + 11 = 347$. ✓All three agree.

Q4 **[8 Marks]**

Explain signed-number representations (sign-magnitude, 1's and 2's complement) and represent +13 and −13 in 8-bit form in each.

Sample Answer. Using a **sign bit** (0 = +, 1 = −): +13 = 00001101 in all three schemes. For −13: **sign-magnitude** = 10001101; **1's complement** = 11110010 (invert +13); **2's complement** = 11110011 (1's complement +1). 2's complement is preferred because subtraction reduces to addition, there is only one zero, and the hardware is simpler.

Q5 **[8 Marks]**

Perform $(25)_{10} - (18)_{10}$ using 8-bit 2's complement, and separately add $(1011)_2 + (1101)_2$, verifying both.

Sample Answer. **Part 1:** -18: $+18 = 00010010$; invert $\rightarrow$ 11101101; $+1 \rightarrow$ 11101110. Add to $+25 = 00011001$: $00011001 + 11101110 = 1\,00000111$; discard carry $\Rightarrow$ $00000111 = 7$, and $25 - 18 = 7$. ✓**Part 2:** $1011 + 1101 = (11000)_2$; check $11 + 13 = 24 = 16 + 8$. ✓

Q6 **[8 Marks]**

Compare the five binary codes (BCD, ASCII, EBCDIC, Unicode, Gray) on purpose, size and typical use, in a table.

Sample Answer.

Code	Type	Size	Use
BCD	Numeric	4 bits/digit	Calculators, meters
ASCII	Alphanumeric	7/8 bits	English text on PCs
EBCDIC	Alphanumeric	8 bits	IBM mainframes
Unicode	Alphanumeric	8–32 bits	All world languages
Gray	Numeric	n bits	Encoders, sensors

BCD codes digits separately; ASCII/EBCDIC/Unicode code text; Gray code minimises bit changes between neighbours. Unicode is the modern standard because it covers every script and is ASCII-compatible via UTF-8.

Q7 **[8 Marks]**

Explain the complete memory hierarchy (registers to secondary storage) with a diagram, comparing speed, cost, size and volatility.

Sample Answer. (Diagram: a pyramid — registers at the top, then cache, main memory (RAM), secondary storage at the base.) **Registers**: inside the CPU, fastest, tiny, volatile. **Cache**: small fast buffer near the CPU, volatile. **RAM**: holds running programs, larger, volatile. **Secondary storage** (SSD/HDD): non-volatile, permanent, large, slowest. Upward: faster, costlier, smaller; downward: cheaper, larger, slower. The hierarchy gives both speed (near the CPU) and capacity (below) economically.

Q8 **[8 Marks]**

Explain the classification of computers by size, purpose and data type, with examples.

Sample Answer. **By size:** supercomputer, mainframe, minicomputer, microcomputer — decreasing power and cost. **By purpose:** general-purpose (a laptop, many tasks) vs special-purpose (an ATM or washing-machine chip, one task). **By data type:** digital (discrete 0/1, most computers), analog (continuous signals, e.g. a speedometer) and hybrid (both, e.g. ICU monitors). These orthogonal classifications describe any computing device.

Q9 **[8 Marks]**

Explain the evolution of programming languages across the five generations, with the translator each needs and their trade-offs.

Sample Answer. **1GL machine** (binary) — no translator, fastest, unreadable. **2GL assembly** (mnemonics) — needs an **assembler**, readable but machine-dependent. **3GL high-level** (C, Python) — needs a **compiler or interpreter**, portable and easy. **4GL** (SQL) — problem-oriented, very high level. **5GL** (Prolog) — declarative/AI, state *what* not *how*. Each generation raises abstraction and portability at a small cost in execution speed.

Q10 **[8 Marks]**

Explain the difference between a compiler and an interpreter in detail, with a diagram and the pros and cons of each.

Sample Answer. A **compiler** reads the whole source program and produces a machine-code file, which then runs (Diagram: Source → Compiler → Executable → Run). An **interpreter** reads and runs one line at a time (Source → Interpreter → Run). Compiler: faster execution, whole-program error report, e.g. C/C++. Interpreter: easier debugging, stops at first error, platform-flexible, e.g. Python. Analogy: translating a whole book (compiler) vs a live translator (interpreter).

Q11 **[8 Marks]**

"A computer is fast and obedient, not wise." Discuss with the characteristics and limitations of computers, giving examples.

Sample Answer. Computers excel at **speed**, **accuracy**, **diligence**, **storage**, **versatility** and **automation** — e.g. sorting a million records in seconds without error. But they have **no intelligence of their own**: they follow instructions literally (*GIGO*), have no common sense or feelings, and cannot fix their own logic errors. Thus the "wisdom" — correct logic and good data — must come from the programmer; the computer merely executes it faithfully.

Q12 **[8 Marks]**

Explain how numbers, text, images and sound are all represented in binary, with one concrete example of each.

Sample Answer. Everything reduces to **0s and 1s**. **Numbers:** positional binary, e.g. $(13)_{10} = 1101$. **Text:** character codes, e.g. 'A' = 65 = 01000001 (ASCII). **Images:** pixels of R,G,B bytes, e.g. red = $(255, 0, 0)$. **Sound:** samples of the wave as numbers (e.g. 44,100 samples/second). Because two voltage levels are easy and reliable to build, all media are encoded, stored and processed as binary patterns.

Q13 **[8 Marks]**

Draw and explain the block diagram of computer organisation showing data flow and control flow separately.

Sample Answer. (Diagram: Input → Memory; Memory ↔ CPU where CPU = CU + ALU; CPU → Output. Solid arrows show **data flow**; dashed arrows from the CU show **control signals** to every unit.) Data moves input → memory → ALU → output, while the **control unit** orchestrates all units by sending timing and control signals. Separating the two flows clarifies that the CU never processes data itself — it only directs.

Q14 **[8 Marks]**

Explain the working of the CPU in detail — CU, ALU, registers and clock — and how they cooperate to run one instruction.

Sample Answer. The **Control Unit** fetches an instruction and decodes it, then signals the other parts. The **ALU** performs the arithmetic $(+, -, \times, \div)$ or logic $(>, <, =$, AND/OR/NOT) required. **Registers** hold the operands and the result during the operation. The **clock** synchronises everything, one step per tick. Example: for `c = a + b`, the CU fetches the ADD instruction, loads a, b into registers, the ALU adds them, and the result register holds c — all within a few clock cycles.

Q15 **[8 Marks]**

Explain data representation of signed and unsigned integers in 8 bits, giving the range of each and the representation of decimal 200 and −56 where valid.

Sample Answer. **Unsigned** 8-bit range: 0 to 255 (2^8 values). **Signed** (2's complement) range: −128 to +127. Decimal 200 is valid only as **unsigned**: 200 = 11001000. Decimal −56 needs **signed** 2's complement: +56 = 00111000; invert → 11000111; +1 → **11001000**. (Note the same bit pattern 11001000 means 200 unsigned but −56 signed — interpretation depends on the chosen scheme.)

Section E — 12 Mark Questions

(Comprehensive multi-part answers; attempt all fifteen)

Q1 **[12 Marks]**

(a) Explain the IPO model and the five functional units with a diagram. (b) Distinguish hardware and software with examples. (c) Justify why understanding hardware helps an AI/ML engineer.

Sample Answer. **(a)** The **IPO model** — input, process, output (with storage) — is realised by five units: input, memory, ALU, control unit, output (diagram: Input → Memory ↔ CPU → Output). **(b)** **Hardware** is physical (CPU, RAM, monitor); **software** is programs (OS, browser); hardware is the body, software the soul. **(c)** AI/ML models are ultimately binary computations on hardware; understanding memory, CPU speed and data representation helps an engineer choose efficient data types, estimate memory/compute needs and debug performance — making them a stronger AI/ML practitioner.

Q2 **[12 Marks]**

Trace the five generations of computers in detail. (a) Give the component, features and one example of each. (b) State the technology change that triggered each new generation. (c) Relate the fifth generation to this AI course.

Sample Answer. **(a)** 1st vacuum tubes (ENIAC), 2nd transistors (IBM 1401), 3rd ICs (System/360), 4th microprocessors (Intel 8086, IBM PC), 5th AI/ULSI (modern AI systems) — smaller, faster, cheaper each time. **(b)** Triggers: transistor (1947) replaced tubes; the IC (1958) packed many transistors; the microprocessor (Intel 4004, 1971) put the CPU on one chip; VLSI then ULSI enabled today's chips. **(c)** The **fifth generation** aims at machines that learn and reason — exactly the AI and Machine Learning studied in Units 4–5 of this course, so this hardware history leads directly into AI.

Q3 [12 Marks]

Data representation. (a) Convert $(423)_{10}$ to binary, octal, hex with steps. (b) Add $(1111)_2 + (0111)_2$. (c) Represent -25 in 8-bit 2's complement and verify by re-conversion.

Sample Answer. **(a)** $423 \div 2 \Rightarrow (110100111)_2$; octal in 3s: $110\,100\,111 = (647)_8$; hex in 4s: $0001\,1010\,0111 = (1A7)_{16}$. Check $(647)_8 = 6{\times}64 + 4{\times}8 + 7 = 423$. ✓ **(b)** $1111 + 0111 = (10110)_2 = 22$, and $15 + 7 = 22$. ✓ **(c)** $+25 = 00011001$; invert $\rightarrow$ 11100110; $+1 \rightarrow$ **11100111**. Re-convert: invert 11100111 $\rightarrow$ 00011000, $+1 \rightarrow 00011001 = 25$, so the value is -25. ✓

Q4 [12 Marks]

Number systems and codes. (a) Explain positional value with an example. (b) Give the binary–octal–hex shortcut with a proof of why it works. (c) Encode $(75)_{10}$ in BCD and in 8-bit binary and contrast them.

Sample Answer. **(a)** Each digit's value = digit $\times$ base$^{\text{position}}$; e.g. $(759)_{10} = 7{\times}10^2 + 5{\times}10 + 9$. **(b)** Since $8 = 2^3$ and $16 = 2^4$, each octal digit maps to exactly 3 bits and each hex digit to 4 bits, so we can group binary digits directly — no division needed. **(c)** $(75)_{10}$ in BCD: $7 = 0111$, $5 = 0101 \Rightarrow 0111\,0101$; in pure 8-bit binary: $75 = 01001011$. They differ because BCD codes each decimal digit separately, while binary encodes the whole value; BCD is easier for digit displays but uses more bits.

Q5 [12 Marks]

(a) Explain the von Neumann architecture with a labelled diagram. (b) Explain the fetch–decode–execute cycle. (c) Define the von Neumann bottleneck and suggest one way modern computers reduce it.

Sample Answer. **(a)** CPU (CU+ALU+registers) $\leftrightarrow$ single memory (program + data) $\leftrightarrow$ I/O, over a shared bus (diagram labelled accordingly). **(b)** The CU *fetches* the next instruction, *decodes* it, and the ALU/registers *execute* it; the cycle repeats each clock tick. **(c)** The **von Neumann bottleneck** is the single shared path between CPU and memory that limits throughput. Modern computers reduce it with **cache memory** (fast local copies) and related techniques, so the CPU waits less for memory.

Q6 [12 Marks]

Signed arithmetic. (a) Explain why 2's complement is preferred. (b) Compute $45 - 60$ in 8-bit 2's complement. (c) Interpret the result's sign bit and re-convert to check.

Sample Answer. **(a)** With **2's complement**, subtraction becomes addition, there is a single representation of zero, and the hardware needs only an adder — simpler and cheaper. **(b)** -60: $+60 = 00111100$; invert $\rightarrow$ 11000011; $+1 \rightarrow 11000100$. Add to $+45 = 00101101$: $00101101 + 11000100 = 11110001$. **(c)** The **sign bit is 1** $\Rightarrow$ negative. Re-convert 11110001: invert $\rightarrow$ 00001110, $+1 \rightarrow 00001111 = 15$, so the value is **-15**, and indeed $45 - 60 = -15$. ✓

Q7 **[12 Marks]**

Programming languages. (a) Classify languages into machine, assembly and high-level with four comparison points. (b) Explain translators (assembler, compiler, interpreter) with a diagram. (c) Justify why Python is used in AI/ML.

Sample Answer. **(a)** **Machine** (binary, fastest, unreadable, machine-dependent); **assembly** (mnemonics, readable, still machine-dependent, needs an assembler); **high-level** (near-English, portable, easy, needs a compiler/interpreter). **(b)** Assembler: assembly $\rightarrow$ machine; compiler: whole HLL $\rightarrow$ machine at once; interpreter: HLL line-by-line (diagram of all three). **(c)** **Python** is used in AI/ML because it is simple and readable, has powerful libraries (NumPy, Pandas, scikit-learn, TensorFlow, PyTorch), and lets engineers focus on the model rather than low-level detail.

Q8 **[12 Marks]**

Memory and storage. (a) Explain the memory hierarchy with a diagram. (b) Distinguish RAM/ROM and primary/secondary memory. (c) A 4 GB RAM stores 32-bit integers; compute how many such integers fit.

Sample Answer. **(a)** Pyramid: registers $\rightarrow$ cache $\rightarrow$ RAM $\rightarrow$ secondary storage; upward faster/costlier/smaller. **(b)** **RAM** is volatile read/write working memory; **ROM** is non-volatile start-up memory. **Primary** memory (RAM/ROM) is directly accessed by the CPU; **secondary** storage (SSD/HDD) is permanent and larger. **(c)** A 32-bit integer = 4 bytes; 4 GB $= 4 \times 1024^3 = 4{,}294{,}967{,}296$ bytes; number of integers $= 4{,}294{,}967{,}296/4 =$ **1,073,741,824** (about 1.07 billion).

Q9 **[12 Marks]**

History of computing. (a) Explain Babbage's contribution and Ada Lovelace's role. (b) Describe ENIAC with two facts. (c) Explain the stored-program concept and why it was revolutionary.

Sample Answer. **(a)** **Charles Babbage** designed the Difference and Analytical Engines; the Analytical Engine (1837) already had a "mill" (CPU), a "store" (memory) and punched-card input. **Ada Lovelace** wrote the first algorithm for it, becoming the first programmer. **(b)** **ENIAC** (1945) used $\sim$18,000 vacuum tubes and weighed $\sim$30 tonnes; it was programmed by re-plugging cables. **(c)** The **stored-program concept** keeps the program in memory with the data, so a new task needs only a new program, not re-wiring — this flexibility is the basis of every modern computer.

Q10 **[12 Marks]**

(a) Explain all five defining characteristics of a computer. (b) State its limitations with examples. (c) Discuss two situations where these limitations matter for AI systems.

Sample Answer. **(a)** **Speed, accuracy, diligence, versatility, storage** (and automation) — e.g. instant, error-free calculation on huge data. **(b)** **No intelligence** (GIGO), no common sense/feelings, cannot self-correct logic — e.g. a mistyped input yields a wrong result silently. **(c)** For AI: (i) **biased or wrong data** produces a biased model (GIGO at scale); (ii) an AI system has **no true understanding**, so it can fail on cases outside its data — hence humans must supply good data and stay in control.

Q11 **[12 Marks]**

Comprehensive conversions. Convert $(0.6875)_{10}$ and $(58)_{10}$ to binary; then combine as $(58.6875)_{10}$ and express in octal and hexadecimal, verifying.

Sample Answer. **Fraction:** $0.6875\times2 = 1.375$ (1), $0.375\times2 = 0.75$ (0), $0.75\times2 = 1.5$ (1), $0.5\times2 = 1.0$ (1) $\Rightarrow$ 0.1011. **Integer:** $58 = 111010$. So $(58.6875)_{10} = (111010.1011)_2$. **Octal** (group 3s each side of the point): $111\,010\,.\,101\,100 = (72.54)_8$. **Hex** (group 4s): $0011\,1010\,.\,1011 = (3A.B)_{16}$. Check integer: $(72)_8 = 58$, $(3A)_{16} = 58$. ✓

Q12 **[12 Marks]**

(a) Explain how a fuzzy of number systems underlies all data. (b) Show that $(1A)_{16} = (32)_8 = (26)_{10} = (11010)_2$. (c) Explain why computers use binary rather than decimal internally.

Sample Answer. **(a)** All data (numbers, text, media) are stored as binary; the octal/hex systems are just compact human-friendly groupings of bits. **(b)** $(1A)_{16} = 1\times16 + 10 = 26$; $(32)_8 = 3\times8 + 2 = 26$; $(11010)_2 = 16 + 8 + 2 = 26$ — all equal $(26)_{10}$. ✓**(c)** Computers use **binary** because electronic switches are naturally two-state (on/off), which is cheap and highly **reliable** against noise; ten clearly separated voltage levels (decimal) would be far harder to build and distinguish.

Q13 **[12 Marks]**

(a) Explain BCD, ASCII and Unicode with examples. (b) Encode "AI" in ASCII binary. (c) Explain why Unicode matters for an AI system serving Indian languages.

Sample Answer. **(a)** **BCD** codes each digit in 4 bits (e.g. 9 = 1001); **ASCII** codes English characters in 7/8 bits (e.g. 'A' = 65); **Unicode** codes *all* scripts (UTF-8, ASCII-compatible). **(b)** 'A' = 65 = 01000001, 'I' = 73 = 01001001 $\Rightarrow$ "AI" = 01000001 01001001. **(c)** An AI system for India must read and generate Hindi, Tamil, Bengali, etc.; only **Unicode** can represent these scripts and emoji, so it is essential for multilingual NLP, chatbots and OCR.

Q14 **[12 Marks]**

(a) Draw the full computer-organisation block diagram. (b) Explain each block's role. (c) Trace what happens, block by block, when a user runs "add two numbers".

Sample Answer. **(a)** Input $\rightarrow$ Memory $\leftrightarrow$ CPU (CU + ALU + registers) $\rightarrow$ Output, with the CU sending control signals (dashed) to all. **(b)** Input takes data in; memory stores program and data; the CU fetches/decodes/directs; the ALU computes; registers hold operands; output shows results. **(c)** The user types a, b (input) $\rightarrow$ stored in memory $\rightarrow$ CU fetches the ADD

instruction and loads a, b into registers → ALU adds them → result stored/displayed (output). This end-to-end trace shows how the units cooperate for even a simple operation.

Q15 **[12 Marks]**

(a) Explain the trade-offs across language levels. (b) Show "multiply two numbers" at each level. (c) Discuss when a programmer should still choose assembly today.

Sample Answer. **(a)** Lower levels (machine/assembly) run faster and give hardware control but are hard to write and non-portable; higher levels are easy and portable at a small speed cost. **(b)** **High-level**: `p = a * b;` **assembly**: `MOV AX,a` / `MUL b;` **machine**: a binary MUL opcode with operands. **(c)** A programmer still chooses **assembly** for embedded systems, device drivers, real-time or safety code, and performance-critical inner loops, where precise control of the hardware and timing outweighs the extra effort.

2 Unit 2 — Visual Thinking and Problem Understanding

Syllabus. **A.** Problem visualisation: mental models vs external representations; benefits for computational thinking. **B.** Visual tools: sketches, mind maps, lists, tables, timelines, storyboards. **C.** Introduction to algorithms and pseudocode; the problem-solving approach. *(CO2–CO3, K2–K3)*

Section A — 2 Mark Questions

(Very short answers; attempt all fifteen)

Q1 **[2 Marks]**

Define problem visualisation in one line.

Sample Answer. **Problem visualisation** is turning a problem into a picture, diagram or structured form so it becomes easier to understand and solve.

Q2 **[2 Marks]**

What is a mental model?

Sample Answer. A **mental model** is the internal picture or idea we build in our mind to understand how something works, letting us predict and reason without the real thing.

Q3 **[2 Marks]**

What is an external representation?

Sample Answer. An **external representation** is a mental model made visible on paper or screen — a sketch, diagram, table or note — that can be shared, checked and revisited.

Q4 **[2 Marks]**

State two advantages of an external representation over a mental model.

Sample Answer. It is **shareable** (others can see it) and **permanent** (it does not fade); it also reduces memory load and helps reveal errors.

Q5 **[2 Marks]**

Name the four pillars of computational thinking.

Sample Answer. **Decomposition**, **pattern recognition**, **abstraction** and **algorithm design**.

Q6 **[2 Marks]**

Define decomposition with one example.

Sample Answer. **Decomposition** means breaking a big problem into smaller, manageable parts; e.g. "make tea" → boil water, add tea, add milk, strain, serve.

Q7 **[2 Marks]**

Define abstraction in computational thinking.

Sample Answer. **Abstraction** means keeping only the essential details of a problem and ignoring the irrelevant ones (e.g. ignoring cup colour when making tea).

Q8 **[2 Marks]**

Name the six visual tools studied in this unit.

Sample Answer. **Sketch**, **mind map**, **list**, **table**, **timeline** and **storyboard**.

Q9 **[2 Marks]**

Which visual tool is best for brainstorming and breaking a topic into parts?

Sample Answer. A **mind map** — the main idea sits in the centre and branches out into sub-ideas, ideal for brainstorming and decomposition.

Q10 **[2 Marks]**

What is the difference between an ordered and an unordered list?

Sample Answer. In an **ordered list** the sequence matters (e.g. recipe steps); in an **unordered list** it does not (e.g. a shopping list).

Q11 **[2 Marks]**

Define an algorithm.

Sample Answer. An **algorithm** is a finite, step-by-step set of unambiguous instructions that solves a problem or completes a task.

Q12 **[2 Marks]**

State any two of Knuth's five properties of an algorithm.

Sample Answer. Any two of: **finiteness**, **definiteness**, **input**, **output**, **effectiveness**.

Q13 **[2 Marks]**

What is pseudocode?

Sample Answer. **Pseudocode** is a way of writing an algorithm using plain, structured English that looks like code but ignores the strict rules of any one programming language.

Q14 **[2 Marks]**

Name the three basic building blocks of any algorithm.

Sample Answer. **Sequence**, **selection** (decision) and **iteration** (loop).

Q15 **[2 Marks]**

Why is "keep adding 1 forever" not a valid algorithm?

Sample Answer. Because it never ends, it violates the **finiteness** property; a valid algorithm must stop after a finite number of steps.

Section B — 4 Mark Questions

(Short answers with an example; attempt all fifteen)

Q1 **[4 Marks]**

Explain why "a picture is worth a thousand words" in problem solving, with an example.

Sample Answer. The human brain grasps **visual structure** far faster than a block of text. For the puzzle "Ravi is taller than Sita; Sita is taller than Meena", a paragraph must be re-read, but a simple vertical diagram (Ravi → Sita → Meena) gives the answer at a glance. The information is identical; only the **representation** changes, and good representation makes thinking easy.

Q2 **[4 Marks]**

Distinguish mental models from external representations on four points.

Sample Answer. **Mental model**: internal, private, fades, high memory load. **External representation**: on paper/screen, shareable, permanent, low memory load. Mental models let us think quickly; external representations let us check, share and refine. They work together — we think with models and externalise them to verify.

Q3 **[4 Marks]**

Explain the four pillars of computational thinking using the example of making tea.

Sample Answer. **Decomposition**: break "make tea" into steps. **Pattern recognition**: "boil water" recurs in coffee/noodles — a reusable sub-task. **Abstraction**: ignore irrelevant details like cup colour. **Algorithm design**: order the steps into a repeatable recipe. The same four pillars solve coding, maths and real-life problems.

Q4 **[4 Marks]**

State four benefits of visualisation for computational thinking.

Sample Answer. **Clarity** — complex ideas become simple; **memory relief** — the paper holds the detail; **error detection** — gaps and mistakes stand out; **communication** — the whole team shares one view. (Also: faster planning before coding.) These make visualisation a core

problem-solving skill.

Q5 **[4 Marks]**

Compare a sketch and a mind map, stating when each is best used.

Sample Answer. A **sketch** is a quick, rough drawing best for *spatial* problems — layouts, shapes, app screens (wireframes). A **mind map** branches sub-ideas from a central topic and is best for *brainstorming* and *decomposition*. Sketches capture "what it looks like"; mind maps capture "what it is made of".

Q6 **[4 Marks]**

When would you use a table versus a timeline? Give one example of each.

Sample Answer. A **table** organises data in rows and columns for *comparison* — e.g. comparing phones by price and battery. A **timeline** places events in *time order* — e.g. a semester schedule or the history of computers. Choose a table to compare attributes and a timeline to show sequence or duration.

Q7 **[4 Marks]**

Explain storyboards and give one situation where they are the right tool.

Sample Answer. A **storyboard** is a sequence of small frames showing what happens step by step, like a comic strip. It is ideal for showing a *flow* of actions or screens over time — e.g. designing an app journey: splash → login → home → profile. It helps a team agree on the "story" before building.

Q8 **[4 Marks]**

List the characteristics of a good algorithm (any four) and briefly explain each.

Sample Answer. **Finiteness** — ends in finite steps; **definiteness** — each step is precise and unambiguous; **input** — zero or more inputs; **output** — at least one output. (Also effectiveness, correctness, generality, efficiency.) Together these ensure the algorithm is clear, terminating and useful.

Q9 **[4 Marks]**

Explain the seven-step problem-solving approach.

Sample Answer. (1) **Understand** the problem; (2) identify **inputs and outputs**; (3) **decompose** into steps; (4) **design** the algorithm; (5) **write** it as pseudocode/flowchart; (6) **test** with examples (dry run); (7) **refine** and fix errors. The habit is: understand, then plan, then code — never jump to code first.

Q10 **[4 Marks]**

Write pseudocode to read two numbers and print their sum, and dry-run it for $a = 4$, $b = 6$.

Sample Answer. `START / INPUT a / INPUT b / SET sum = a + b / PRINT sum / STOP`. Dry run: $a = 4$, $b = 6 \Rightarrow sum = 10$; output **10**. This is a pure *sequence* algorithm.

Q11 **[4 Marks]**

Write pseudocode to find the larger of two numbers and dry-run it for $a = 2$, $b = 9$.

Sample Answer. `START / INPUT a, b / IF a > b THEN PRINT a ELSE PRINT b / STOP`. For $a = 2$, $b = 9$: is $2 > 9$? No $\Rightarrow$ print **9**. This uses *selection* (a decision).

Q12 **[4 Marks]**

Differentiate an algorithm, pseudocode and a program.

Sample Answer. An **algorithm** is the plan in any form; **pseudocode** is that plan written code-like in plain English; a **program** is the plan written in a real language (Python, C) that a computer can run. Order of use: algorithm $\rightarrow$ pseudocode $\rightarrow$ program.

Q13 **[4 Marks]**

Explain sequence, selection and iteration with a one-line example of each.

Sample Answer. **Sequence**: steps in order — "read a, read b, add". **Selection**: choose a path — "IF marks $\geq$ 40 THEN Pass ELSE Fail". **Iteration**: repeat — "FOR $i = 1$ TO 5 print i". Every program, however large, is built from just these three structures.

Q14 **[4 Marks]**

Explain the role of representation in problem solving with a real example.

Sample Answer. The *form* in which we state a problem affects how easily we solve it. Written directions are hard to follow, but a **map** of the same route is instantly clear; a wall of numbers is confusing, but a **table** or **chart** reveals the pattern. Choosing a good representation is often the first real step toward the solution.

Q15 **[4 Marks]**

Why is computational thinking useful beyond programming? Give two non-computing examples.

Sample Answer. **Computational thinking** is a general skill for solving problems clearly and step by step. Examples: planning a wedding (decompose into venue, food, guests; reuse checklists — pattern recognition); organising a library (abstraction: categorise by subject; algorithm: a shelving rule). It applies to science, business and daily life, not only coding.

Section C — 6 Mark Questions

(Explanation with a small diagram or a two-step example; attempt all fifteen)

Q1 **[6 Marks]**

Explain, with a diagram, how externalising a mental model reduces cognitive load.

Sample Answer. A **mental model** lives in the mind, where it is limited by memory and easily forgotten. **Externalising** it — drawing it on paper — moves the storage burden off the brain (Diagram: Mind $\xrightarrow{\text{draw it}}$ Paper/Screen). The paper now "remembers" the details, freeing working memory to *reason* rather than just *hold* information. This is why we sketch, list and tabulate while solving hard problems.

Q2 **[6 Marks]**

Compare all six visual tools in a table (best-for and one example each).

Sample Answer.

Tool	Best for	Example
Sketch	Shapes, layouts	App screen
Mind map	Brainstorming	Project plan
List	Steps/checklists	Recipe steps
Table	Comparison	Phone comparison
Timeline	Order of events	Semester plan
Storyboard	Flow of actions	App journey

Many real problems combine several — a mind map to plan, then a list to order the steps.

Q3 **[6 Marks]**

Draw a mind map (described) for planning a college trip and explain its structure.

Sample Answer. Centre: **Plan a Trip**. Main branches: **Budget** (food, tickets), **Transport** (bus, train), **Stay** (hotel, hostel), **Places** (temples, market). Each branch splits into sub-branches. The radial structure mirrors **decomposition**: the big task at the centre, smaller tasks around it, and details at the leaves — making the whole plan visible at once.

Q4 **[6 Marks]**

Write pseudocode to compute the sum of the first n natural numbers and dry-run it for $n = 4$.

Sample Answer. `START / INPUT n / SET sum=0 / FOR i=1 TO n DO sum=sum+i / ENDFOR / PRINT sum / STOP`. Dry run for $n = 4$: i=1:sum=1; i=2:sum=3; i=3:sum=6; i=4:sum=10. Output **10**. A loop needs three parts: initialise ($i = 1$), test ($i \leq n$), update ($i = i + 1$).

Q5 **[6 Marks]**

Explain the characteristics of a good algorithm and test whether "divide by the count of numbers" is well defined when the count is zero.

Sample Answer. A good algorithm has **finiteness**, **definiteness**, **input**, **output** and **effectiveness**. "Divide by the count" fails **definiteness/effectiveness** when the count is 0, because division by zero is undefined and cannot be carried out. A correct algorithm must first check "if count = 0 then report empty", showing why precise, executable steps matter.

Q6 **[6 Marks]**

Write pseudocode to check whether a number is even or odd, dry-run it for 7 and 10, and name the control structure used.

Sample Answer. `START / INPUT n / IF n MOD 2 = 0 THEN PRINT "Even" ELSE PRINT "Odd" / STOP`. For 7: $7 \bmod 2 = 1 \Rightarrow$ **Odd**; for 10: $10 \bmod 2 = 0 \Rightarrow$ **Even**. The control structure is **selection** (an IF–ELSE decision).

Q7 **[6 Marks]**

Explain decomposition and pattern recognition together, using the example of building a food-delivery app.

Sample Answer. **Decomposition** splits the app into parts: login, restaurant list, cart, payment, tracking. **Pattern recognition** spots repeated sub-problems: "show a list of items" appears for restaurants, dishes and past orders, so one reusable component solves all three. Together they turn a huge task into small, partly reusable pieces — the essence of good software design.

Q8 **[6 Marks]**

Convert this word problem into an ordered list and then into pseudocode: "To log in, open the app, type the username, type the password, and tap Sign in."

Sample Answer. **Ordered list:** 1. Open app; 2. Type username; 3. Type password; 4. Tap Sign in. **Pseudocode:** `START / OPEN app / INPUT username / INPUT password / IF valid THEN PRINT "Welcome" ELSE PRINT "Try again" / STOP`. The ordered list is the seed of the algorithm — a sequence of steps in the right order.

Q9 **[6 Marks]**

Explain, with a small example, why the same data is clearer as a table than as prose.

Sample Answer. Prose: "Phone A costs 12k with 4000 mAh; B costs 18k with 5000 mAh; C costs 22k with 4500 mAh." This is hard to compare. As a **table** (rows A/B/C, columns Price/Battery) the eye scans each column instantly and spots that B has the largest battery. Tables expose **patterns** and support **comparison** that prose hides.

Q10 **[6 Marks]**

Explain the problem-solving approach with a worked example (finding the average of three numbers).

Sample Answer. **Understand**: find the mean of three numbers. **Inputs/outputs**: inputs a, b, c; output average. **Decompose**: read, add, divide, print. **Design + pseudocode**: `INPUT a,b,c / avg=(a+b+c)/3 / PRINT avg`. **Test**: a=10, b=20, c=30 $\Rightarrow$ avg=20. **Refine**: guard against division by a wrong count. This shows understand → plan → test before coding.

Q11 **[6 Marks]**

Draw (describe) a storyboard for an online exam and explain each frame.

Sample Answer. Frames: (1) **Login** — student signs in; (2) **Instructions** — rules shown; (3) **Question screen** — one question with a timer; (4) **Review** — flagged questions; (5) **Submit** — confirmation. Arrows join the frames left to right. The storyboard shows the *flow* of screens over time, so designers and students agree on the journey before the app is built.

Q12 **[6 Marks]**

Explain how visual tools support each pillar of computational thinking, with one tool per pillar.

Sample Answer. **Decomposition** → a **mind map** splits the problem into branches. **Pattern recognition** → a **table** lines up cases so repeats stand out. **Abstraction** → a **sketch** keeps the essential shape and drops detail. **Algorithm design** → a **list or flowchart** orders the steps. Thus visuals make each thinking step concrete and easier.

Q13 **[6 Marks]**

Write pseudocode to find the largest of three numbers and dry-run it for $(3, 9, 5)$.

Sample Answer. `START / INPUT a,b,c / SET max=a / IF b>max THEN max=b / IF c>max THEN max=c / PRINT max / STOP`. Dry run $(3, 9, 5)$: max=3; b=9 $>$ 3 $\Rightarrow$ max=9; c=5 $>$ 9? No; output **9**. The pattern "compare and update a running maximum" is common in programming.

Q14 **[6 Marks]**

Explain the difference between a flowchart and pseudocode as two representations of the same algorithm.

Sample Answer. Both describe the *same* logic in different forms. **Pseudocode** is *text*: structured English statements read top to bottom. A **flowchart** is a *picture*: standard symbols (oval, rectangle, diamond) joined by arrows. Text is compact and quick to write; the picture makes branching and loops visually obvious. Choosing between them depends on the audience and the complexity.

Q15 **[6 Marks]**

Given the list $[4, 9, 2, 7]$, write and dry-run pseudocode to find its maximum using a state table.

Sample Answer. `max = first element; FOR each item: IF item > max THEN max = item`. State table: start max=4; see 9 $\Rightarrow$ max=9; see 2 (no change); see 7 (no change). Output

9. Tracking `max` step by step (a **state table**) confirms the logic before coding.

Section D — 8 Mark Questions (Full explanation with diagram or complete example; attempt all fifteen)

Q1 **[8 Marks]**

Explain in detail the interplay between mental models and external representations in solving a complex problem, with a diagram.

Sample Answer. We first form a **mental model** — an internal picture of the problem — which lets us reason quickly but is private, fuzzy and easily forgotten. To manage complexity we **externalise** it as a sketch, table or diagram (Diagram: Mind $\xrightarrow{\text{draw}}$ Paper $\xrightarrow{\text{read}}$ Mind, a loop). The external form is shareable, permanent and low-load, so it exposes errors and lets a team collaborate; reading it back *refines* the mental model. Thus the two are complementary: thinking happens in the mind, but hard thinking is offloaded to paper and cycled back — the foundation of good problem understanding.

Q2 **[8 Marks]**

Explain the four pillars of computational thinking in detail, with a real software example for each.

Sample Answer. **Decomposition** — split a chat app into login, contacts, messaging, media. **Pattern recognition** — a "scrollable list" recurs for chats, contacts and media, so one component is reused. **Abstraction** — model a "message" by sender, text and time, ignoring font or pixel detail. **Algorithm design** — define the exact steps to send, deliver and display a message. Together they convert a large, vague requirement into small, reusable, precisely specified pieces — how real systems are built, and the mindset behind all of AI/ML engineering.

Q3 **[8 Marks]**

Explain the complete problem-solving approach and apply every step to "find whether a year is a leap year".

Sample Answer. **Understand**: a leap year is divisible by 4, except centuries not divisible by 400. **Inputs/outputs**: input year; output leap/not. **Decompose**: check $\div 4$, $\div 100$, $\div 400$. **Design/pseudocode**: `IF (year MOD 400 = 0) OR (year MOD 4 = 0 AND year MOD 100 ≠ 0) THEN leap ELSE not-leap`. **Test**: 2000 → leap; 1900 → not; 2024 → leap; 2023 → not. **Refine**: handle invalid input. This shows each step from understanding to a tested rule.

Q4 **[8 Marks]**

Write and fully dry-run pseudocode that prints the multiplication table of a number N (1 to 10), using a state table for $N = 3$.

Sample Answer. `START / INPUT N / FOR i=1 TO 10 DO PRINT N "x" i "=" N*i / ENDFOR / STOP`. State table for $N = 3$: i=1→3, i=2→6, i=3→9, …, i=10→30. Output: the ten lines $3, 6, 9, \ldots, 30$. The loop repeats one PRINT step ten times; tracking i and $N \times i$

confirms correctness before coding.

Q5 **[8 Marks]**

Discuss the six visual tools in depth: for each, give its structure, one strength, and one real use-case.

Sample Answer. **Sketch** — rough drawing; strength: fast; use: wireframing an app screen. **Mind map** — radial branches; strength: brainstorming; use: planning a project. **List** — linear items; strength: ordering steps; use: a checklist. **Table** — rows/columns; strength: comparison; use: comparing laptops. **Timeline** — events on a line; strength: sequence/duration; use: a project schedule. **Storyboard** — frames; strength: showing flow; use: an app user journey. Skilled problem-solvers pick and combine these deliberately.

Q6 **[8 Marks]**

Explain algorithms fully: definition, characteristics, and how they relate to pseudocode and programs, with an example carried through all three forms.

Sample Answer. An **algorithm** is a finite sequence of unambiguous steps with the properties finiteness, definiteness, input, output, effectiveness. Example "add two numbers": **Algorithm** (steps): read a, b; add; print. **Pseudocode**: `INPUT a,b / sum=a+b / PRINT sum`. **Program** (Python): `a=int(input()); b=int(input()); print(a+b)`. The algorithm is language-free; pseudocode expresses it readably; the program makes it runnable. Designing the algorithm first prevents bugs later.

Q7 **[8 Marks]**

Explain the three control structures with a flowchart-style description and a combined example that uses all three.

Sample Answer. **Sequence** runs steps in order; **selection** branches on a condition; **iteration** repeats. Combined example — "print whether each of n numbers is even": `FOR i=1 TO n (iteration) DO READ x (sequence); IF x MOD 2 = 0 (selection) THEN PRINT ''even'' ELSE PRINT ''odd''`. The flowchart would show a loop box containing a decision diamond. Any program is a nesting of these three structures.

Q8 **[8 Marks]**

A student says "I'll just start coding." Argue, with an example, why understanding and visualising first produces better results.

Sample Answer. Jumping to code risks solving the *wrong* problem or missing cases. Example: asked for "average marks of passed students", a coder who skips understanding may average *all* students. **Understanding** clarifies "passed only"; **visualising** (a flowchart with a filter step) exposes the needed condition; a **dry run** on sample data catches errors early. Thus understand → visualise → plan → code yields correct, maintainable solutions, while code-first often wastes time in debugging.

Q9 **[8 Marks]**

Explain how tables and timelines each reveal structure that prose hides, with a worked example of each.

Sample Answer. **Table** example: three courses with credits and marks — as prose it is a jumble, but a rows/columns table lets you total credits and compare marks by scanning a column. **Timeline** example: project tasks with dates — prose obscures overlaps, but a timeline shows which tasks run in parallel and where the deadline pressure is. Both convert linear text into a spatial layout, so the eye extracts comparisons (table) and sequence/duration (timeline) instantly.

Q10 **[8 Marks]**

Design (in pseudocode) and dry-run an algorithm that counts how many numbers in a list are positive, negative and zero, for the list $[3, -2, 0, 5, -1]$.

Sample Answer. `SET pos=0,neg=0,zero=0 / FOR each x: IF x>0 THEN pos++ ELSE IF x<0 THEN neg++ ELSE zero++`. Dry run: $3 \to pos1$; $-2 \to neg1$; $0 \to zero1$; $5 \to pos2$; $-1 \to neg2$. Result: **pos=2, neg=2, zero=1**. The state table (three counters updated per item) verifies the branching logic before implementation.

Q11 **[8 Marks]**

Explain, with examples, how the choice of representation can make a hard problem easy or an easy problem hard.

Sample Answer. Good representation aids thinking; poor representation obstructs it. The "taller than" puzzle is trivial as a vertical diagram but confusing as prose (easy made easy). Roman numerals make multiplication painful, while positional decimals make it easy (hard made easy by representation). Conversely, forcing a naturally visual routing problem into a text list hides the structure (easy made hard). Hence choosing the right representation is a genuine problem-solving skill.

Q12 **[8 Marks]**

Explain the full life cycle from a real-world problem to a tested pseudocode solution, using "withdraw cash from an ATM".

Sample Answer. **Understand**: dispense requested cash if the account and balance allow. **Inputs/outputs**: card, PIN, amount; output cash or error. **Decompose**: authenticate, check balance, dispense, update. **Pseudocode**: `IF PIN correct THEN IF balance` $\geq$ `amount THEN dispense; balance = balance − amount ELSE "insufficient" ELSE "wrong PIN"`. **Test** cases: correct PIN + enough balance (success); enough vs insufficient balance (boundary); wrong PIN (error). This end-to-end trace shows decomposition, selection and testing working together.

Q13 **[8 Marks]**

Discuss the benefits and limitations of visual thinking, and when a purely textual/algebraic approach is better.

Sample Answer. **Benefits**: clarity, memory relief, error-spotting, communication, faster planning. **Limitations**: complex diagrams can themselves become cluttered; some problems (precise numerical or symbolic ones) are better handled algebraically. For example, solving a quadratic equation is faster with the formula than with a drawing, while designing a user journey is far clearer as a storyboard. A good engineer matches the tool — visual, textual or algebraic — to the problem.

Q14 **[8 Marks]**

Explain nested selection with a flowchart-style description and write pseudocode that assigns grades A/B/C/F from marks.

Sample Answer. **Nested selection** is a decision inside a decision (an else-if ladder). Pseudocode: `IF m≥90 THEN A ELSE IF m≥80 THEN B ELSE IF m≥70 THEN C ELSE F`. The flowchart is a chain of diamonds, each "No" leading to the next test. Dry run: m=85 ⇒ B; m=72 ⇒ C; m=50 ⇒ F. Nested selection handles many mutually exclusive cases cleanly.

Q15 **[8 Marks]**

Explain the whole "understand–visualise–plan–test" pipeline using "count the vowels in a word", giving pseudocode and a dry run for "AIML".

Sample Answer. **Understand**: count how many letters are vowels (a, e, i, o, u). **Visualise/plan**: loop over each character, check membership in the vowel set, keep a counter. **Pseudocode**: `count=0 / FOR each ch in word: IF ch is a vowel THEN count=count+1 / PRINT count`. **Dry run** for "AIML": A(vowel→1), I(vowel→2), M(no), L(no) ⇒ **2**. **Test** cases: "AEIOU" → 5, "XYZ" → 0, "" (empty) → 0. This shows the full pipeline from understanding a problem to a tested solution — the habit this unit builds.

Section E — 12 Mark Questions

(Comprehensive multi-part answers; attempt all fifteen)

Q1 **[12 Marks]**

(a) Define problem visualisation and explain its benefits. (b) Distinguish mental models and external representations with a diagram. (c) Discuss why this matters for an AI/ML engineer.

Sample Answer. **(a)** **Problem visualisation** turns a problem into a picture; benefits: clarity, memory relief, error detection, communication, faster planning. **(b)** **Mental model** (internal, private, fades) vs **external representation** (on paper, shareable, permanent) — diagram: Mind ↔ Paper. **(c)** AI/ML work involves complex data pipelines and model logic; visualising data distributions, model architectures and decision boundaries helps an engineer understand, debug and communicate their work — making visual thinking a core professional skill, not just a study aid.

Q2 **[12 Marks]**

(a) Explain the four pillars of computational thinking. (b) Apply all four to designing a simple library-management system. (c) State how each pillar reappears in machine learning.

Sample Answer. **(a)** Decomposition, pattern recognition, abstraction, algorithm design. **(b)** *Decompose*: catalogue, issue, return, fines. *Patterns*: "search a list" recurs for books and members. *Abstraction*: a "book" = title, author, ID, status. *Algorithm*: steps to issue a book with checks. **(c)** In ML: decomposition → splitting a task into data/model/eval; patterns → the model *learns* patterns; abstraction → features summarise raw data; algorithm design → the training procedure. Thus this unit's thinking directly underpins AI.

Q3 **[12 Marks]**

(a) Define an algorithm and its five properties. (b) Write pseudocode to compute the factorial of n. (c) Dry-run it for $n = 4$ with a state table and state which property guarantees it stops.

Sample Answer. **(a)** An algorithm is a finite sequence of unambiguous steps with **finiteness, definiteness, input, output, effectiveness**. **(b)** `INPUT n / SET fact=1 / FOR i=1 TO n DO fact=fact*i / ENDFOR / PRINT fact`. **(c)** State table $n = 4$: i=1→$fact$1, i=2→2, i=3→6, i=4→24; output **24**. The loop runs a fixed n times, so **finiteness** guarantees termination.

Q4 **[12 Marks]**

(a) Explain the six visual tools with structure and use. (b) For a "plan a college fest" problem, choose and justify three tools. (c) Show a small mind map and a small ordered list for the fest.

Sample Answer. **(a)** Sketch (spatial), mind map (brainstorm), list (order), table (compare), timeline (schedule), storyboard (flow). **(b)** For a fest: a **mind map** to brainstorm all areas; a **timeline** for the day's schedule; a **table** to compare vendor quotes. **(c)** Mind map centre "Fest" with branches Events, Food, Sponsors, Publicity. Ordered list for the day: 1. Inauguration; 2. Competitions; 3. Lunch; 4. Cultural show; 5. Prize distribution. Together they plan the fest from ideas to a timed schedule.

Q5 **[12 Marks]**

(a) Explain the problem-solving approach. (b) Fully solve "check whether a number is prime" through all steps. (c) Design three test cases and justify each.

Sample Answer. **(a)** Understand → I/O → decompose → design → pseudocode → test → refine. **(b)** A prime > 1 has no divisor other than 1 and itself. `INPUT n / IF n≤1 THEN "not prime" / FOR i=2 TO n-1 DO IF n MOD i = 0 THEN "not prime"; STOP / PRINT "prime"`. **(c)** Tests: 7 (normal prime); 9 (normal composite); 1 (boundary — not prime by definition). Each targets a different behaviour, so passing all three gives confidence in the logic.

Q6 **[12 Marks]**

(a) Explain sequence, selection and iteration. (b) Write pseudocode using all three to print the sum of even numbers from 1 to n. (c) Dry-run it for $n = 6$.

Sample Answer. **(a)** Sequence (in order), selection (decide), iteration (repeat). **(b)** `INPUT n / SET sum=0 / FOR i=1 TO n DO (IF i MOD 2 = 0 THEN sum=sum+i) / ENDFOR / PRINT sum`. **(c)** $n = 6$: even i are $2, 4, 6$; $sum = 2 \to 6 \to 12$. Output **12**. The loop (iteration) contains a decision (selection) and assignments (sequence) — all three structures in one small program.

Q7 [12 Marks]

(a) Explain how representation affects difficulty. (b) Give two problems that become easy with the right visual. (c) Give one problem where a visual is unhelpful and explain why.

Sample Answer. **(a)** The form of a problem changes how hard it is to think about; a good representation exposes structure. **(b)** A family-tree question is easy as a **tree diagram**; a scheduling clash is easy on a **timeline**. **(c)** Computing a precise compound-interest value is better done **algebraically** with the formula — a drawing adds nothing and hides the exact arithmetic. Hence match the representation to the problem: visual for structure, symbolic for precise computation.

Q8 [12 Marks]

(a) Explain decomposition and abstraction in depth. (b) Apply them to model a "student" and a "course" for a result system. (c) Discuss how abstraction connects to features in machine learning.

Sample Answer. **(a)** **Decomposition** breaks a system into parts; **abstraction** keeps essentials and hides detail. **(b)** A **student** = (ID, name, marks); a **course** = (code, title, credits) — irrelevant details are dropped. The result system decomposes into enrol, record marks, compute GPA. **(c)** In ML, **features** are exactly abstractions of raw data — e.g. representing an email by "number of links, has the word 'free', sender reputation". Choosing good features is applied abstraction, and it strongly affects model accuracy.

Q9 [12 Marks]

(a) Write pseudocode to search for a value in a list (linear search). (b) Dry-run it on $[5, 8, 2, 9]$ searching for 2. (c) Design test cases including the not-found case.

Sample Answer. **(a)** `INPUT list, key / SET found=NO / FOR each item WITH index i: IF item = key THEN found=YES; position=i; STOP / IF found THEN PRINT position ELSE PRINT "not found"`. **(b)** $[5, 8, 2, 9]$, key 2: compare 5(no), 8(no), 2(yes) $\Rightarrow$ position **3**. **(c)** Tests: key present at start (5), key present at end (9), key absent ($7 \to$ "not found"). These cover normal, boundary and not-found cases, verifying the search fully.

Q10 [12 Marks]

(a) Explain the benefits and limitations of visual thinking. (b) Compare a flowchart and pseudocode for the same task. (c) Recommend which to use for teaching beginners and justify.

Sample Answer. **(a)** Benefits: clarity, memory relief, error-spotting, communication; limitations: clutter for large problems, weaker for precise symbolic work. **(b)** For "larger of two numbers", a **flowchart** shows the branch as a diamond with Yes/No arrows, while **pseudocode**

states `IF a>b THEN a ELSE b` — same logic, picture vs text. **(c)** For **beginners**, start with flowcharts: the visual branching and looping build intuition; then move to pseudocode for compactness. Combining both cements understanding before real coding.

Q11 **[12 Marks]**

(a) Explain the seven-step approach with the "simple calculator" problem. (b) Give its pseudocode. (c) Dry-run addition and division-by-zero cases.

Sample Answer. **(a)** Understand (do $+, -, \times, \div$ on two numbers) → I/O → decompose by operator → design → pseudocode → test → refine. **(b)** `INPUT a,b,op / IF op=+ THEN r=a+b / ELSE IF op=- THEN r=a-b / ELSE IF op=x THEN r=a*b / ELSE IF op=/ THEN (IF b≠0 THEN r=a/b ELSE "error") / PRINT r`. **(c)** a=6, b=3, + ⇒ 9; a=6, b=0, ÷ ⇒ **error**. The refine step's zero check prevents a runtime crash.

Q12 **[12 Marks]**

(a) Explain how mind maps aid decomposition. (b) Build a mind map (described) for "healthy living". (c) Convert one branch into an ordered checklist.

Sample Answer. **(a)** A **mind map** places the whole topic at the centre and forces you to name its parts as branches — literally performing **decomposition** on paper. **(b)** Centre "Healthy Living"; branches: **Diet** (balanced meals, water), **Exercise** (walking, yoga), **Sleep** (7–8 hrs), **Mind** (breaks, hobbies). **(c)** Checklist for *Diet*: 1. Eat vegetables daily; 2. Drink 2–3 L water; 3. Limit junk food; 4. Fixed meal times. The map generates ideas; the checklist orders one branch into action.

Q13 **[12 Marks]**

(a) Explain state tables and variable tracking. (b) Dry-run a swap of two variables using a temporary variable. (c) Explain the danger of swapping without a temporary.

Sample Answer. **(a)** A **state table** has a column per variable and a row per step; you record each variable's value as it changes (**variable tracking**). **(b)** Swap a=5, b=8: `temp=a`→temp5; `a=b`→a8; `b=temp` →b5. Result a=8, b=5. **(c)** Without a **temporary**, writing `a=b` first overwrites a, so its original value is lost and both variables end up equal to b. The temporary preserves the value — a classic case where tracing reveals a subtle bug.

Q14 **[12 Marks]**

(a) Explain the role of test cases. (b) Design normal, boundary and invalid test cases for "classify BMI". (c) Explain how tracing plus test cases catch logic errors.

Sample Answer. **(a)** A **test case** pairs an input with its expected output; passing many cases builds confidence. **(b)** For BMI bands: *normal* BMI 22 → "normal"; *boundary* BMI exactly 25 → the exact band edge; *invalid* negative height → "error". **(c)** A **dry run** shows how variables flow, revealing wrong comparisons (e.g. using $>$ instead of $\geq$ at a boundary), and **test cases** confirm the fix. Together they catch **logic errors** that a compiler cannot detect — exactly the skill Unit 3 develops next.

Q15 **[12 Marks]**

(a) Summarise the whole unit. (b) Explain how visualisation, computational thinking, tools and pseudocode form one pipeline. (c) State how this prepares you for Units 3–5.

Sample Answer. **(a)** We learned to **visualise** problems, apply **computational thinking** (decompose, patterns, abstraction, algorithm design), use six **visual tools**, and write **algorithms/pseudocode**. **(b)** These form one pipeline: understand → visualise → decompose → design → pseudocode → test. **(c)** Unit 3 turns this logic into **flowcharts, tables and dry runs**; Unit 4 introduces **AI**; Unit 5 teaches machines to **learn**. Solid problem understanding here is the foundation for all of them.

3 Unit 3 — Visualizing Processes and Logic

Syllabus. A. Flowchart fundamentals and symbols; control structures (sequence, selection, iteration). **B.** Decision tables and truth tables for logical conditions. **C.** Dry run and execution tracing; state tables; test-case design and debugging. *(CO3, K2–K4)*

Section A — 2 Mark Questions

(Very short answers; attempt all fifteen)

Q1 **[2 Marks]**

What is a flowchart?

Sample Answer. A **flowchart** is a diagram that represents the steps of an algorithm or process using standard symbols joined by arrows showing the flow of control.

Q2 **[2 Marks]**

Name the flowchart symbol used for start/stop and its shape.

Sample Answer. The **terminal** symbol, drawn as an **oval** (rounded rectangle), marks the start and stop of a flowchart.

Q3 **[2 Marks]**

Which flowchart symbol represents a decision, and what shape is it?

Sample Answer. The **decision** symbol, drawn as a **diamond (rhombus)**, represents a yes/no or true/false test.

Q4 **[2 Marks]**

Which symbol represents input/output in a flowchart?

Sample Answer. The **parallelogram** represents **input/output** operations (reading data or printing results).

Q5 **[2 Marks]**

Name the three basic control structures.

Sample Answer. **Sequence**, **selection** (branching) and **iteration** (looping).

Q6 **[2 Marks]**

What is a decision table?

Sample Answer. A **decision table** is a tabular way of showing conditions, their possible combinations, and the actions to take for each combination.

Q7 **[2 Marks]**

What is a truth table?

Sample Answer. A **truth table** lists every possible combination of truth values (T/F) of the inputs and the resulting output of a logical expression.

Q8 **[2 Marks]**

How many rows does a truth table with 3 input variables have?

Sample Answer. $2^3 = 8$ rows, since each of the 3 variables can be T or F.

Q9 **[2 Marks]**

What is a dry run?

Sample Answer. A **dry run** is manually tracing an algorithm step by step on paper, tracking variable values, to check its logic before running it on a computer.

Q10 **[2 Marks]**

What is a state table (trace table)?

Sample Answer. A **state table** is a table with a column for each variable and a row for each step, recording how the variables change during a dry run.

Q11 **[2 Marks]**

What is a test case?

Sample Answer. A **test case** is a specific input together with its expected output, used to check whether a program behaves correctly.

Q12 **[2 Marks]**

Give the truth values of AND for inputs (T, F) and (T, T).

Sample Answer. AND is true only when both inputs are true: $(T, F) \Rightarrow$ **F**; $(T, T) \Rightarrow$ **T**.

Q13 **[2 Marks]**

What is the result of the logical OR of (F, F) and (F, T)?

Sample Answer. OR is true if at least one input is true: $(F, F) \Rightarrow$ **F**; $(F, T) \Rightarrow$ **T**.

Q14 **[2 Marks]**

What is debugging?

Sample Answer. **Debugging** is the process of finding and fixing errors (bugs) in an algorithm or program.

Q15 **[2 Marks]**

What is a boundary test case? Give one example.

Sample Answer. A **boundary test case** checks values at the edge of a valid range; e.g. for "pass if marks ≥ 40", testing exactly **40** checks the boundary.

Section B — 4 Mark Questions

(Short answers with an example; attempt all fifteen)

Q1 **[4 Marks]**

List the standard flowchart symbols and state the purpose of each.

Sample Answer. **Oval** (terminal) — start/stop; **parallelogram** — input/output; **rectangle** — process/action; **diamond** — decision; **arrow** — flow of control; **circle** — connector (joins parts). Using standard symbols makes a flowchart readable by anyone who knows the convention.

Q2 **[4 Marks]**

State the rules/guidelines for drawing a good flowchart (any four).

Sample Answer. (1) Use **standard symbols**; (2) flow generally **top-to-bottom, left-to-right**; (3) every flowchart has exactly one **start** and at least one **stop**; (4) a decision has **labelled** branches (Yes/No). (Also: keep it simple, avoid crossing lines, use connectors for long charts.) These keep the chart clear and unambiguous.

Q3 **[4 Marks]**

Draw (describe) a flowchart to check whether a number is positive or negative.

Sample Answer. **Start** → input n (parallelogram) → decision diamond "$n \geq 0$?": **Yes** → print "Positive"; **No** → print "Negative" → **Stop**. The single diamond gives two exits, illustrating **selection**. (Zero is treated as non-negative here.)

Q4 **[4 Marks]**

Explain the three control structures with a one-line flowchart description of each.

Sample Answer. **Sequence**: boxes stacked top to bottom, one after another. **Selection**: a diamond with Yes/No branches that rejoin. **Iteration**: a loop where a decision sends control back to an earlier step until a condition is met. Every process reduces to these three patterns.

Q5 **[4 Marks]**

Construct the truth table for A AND B.

Sample Answer.

A	B	$A \wedge B$
F	F	F
F	T	F
T	F	F
T	T	T

AND outputs true **only** when both inputs are true — the last row.

Q6 **[4 Marks]**

Construct the truth table for A OR B.

Sample Answer.

A	B	$A \vee B$
F	F	F
F	T	T
T	F	T
T	T	T

OR outputs true when **at least one** input is true — false only in the first row.

Q7 **[4 Marks]**

Construct the truth table for NOT A and explain the NOT operator.

Sample Answer.

A	$\neg A$
F	T
T	F

NOT is a unary operator that **inverts** its input: true becomes false and false becomes true.

Q8 **[4 Marks]**

Build a simple decision table for "grant a loan if income is high AND credit is good".

Sample Answer.

High income?	Good credit?	Action
Y	Y	Grant
Y	N	Reject
N	Y	Reject
N	N	Reject

The table enumerates all four condition combinations; only the first grants the loan, matching

the AND rule.

Q9 **[4 Marks]**

Perform a dry run of: `sum=0; FOR i=1 TO 3 DO sum=sum+i`. Show the state table.

Sample Answer.

Step	i	sum
init	–	0
1	1	1
2	2	3
3	3	6

Final **sum** $= 6$. The state table tracks each variable per iteration.

Q10 **[4 Marks]**

Explain the difference between a syntax error and a logic error with an example.

Sample Answer. A **syntax error** breaks the language's grammar and is caught by the compiler/interpreter (e.g. a missing bracket). A **logic error** is grammatically valid but produces wrong results (e.g. using + instead of −); it runs but gives the wrong answer. Logic errors are found by **dry runs** and **test cases**, not by the compiler.

Q11 **[4 Marks]**

Design three test cases (normal, boundary, invalid) for "pass if marks ≥ 40".

Sample Answer. **Normal**: marks = 75 → Pass. **Boundary**: marks = 40 → Pass (edge of the rule). **Invalid**: marks = −5 → error (marks cannot be negative). Testing the boundary catches the common >-vs-≥ mistake.

Q12 **[4 Marks]**

Explain the purpose of a connector symbol in a flowchart with an example.

Sample Answer. A **connector** (small circle with a label) joins parts of a flowchart that cannot be linked directly — e.g. when a chart continues on the next page or another column. An "A" connector at the break matches an "A" where the flow resumes, avoiding long or crossing arrows and keeping the chart tidy.

Q13 **[4 Marks]**

Write and trace pseudocode using a WHILE loop to print numbers 1 to 3.

Sample Answer. `i=1 / WHILE i≤3 DO PRINT i; i=i+1 / ENDWHILE`. Trace: i=1 print 1, i=2 print 2, i=3 print 3, i=4 stops. Output **1 2 3**. A WHILE loop tests *before* each pass, so it may run zero times if the condition is false initially.

Q14 **[4 Marks]**

Distinguish between a pre-test (WHILE) and a post-test (DO–WHILE) loop.

Sample Answer. A **WHILE** loop checks the condition *before* the body, so it can execute **zero** times. A **DO–WHILE** loop checks *after* the body, so it always executes **at least once**. Choose DO–WHILE when the action must happen before the first check (e.g. ask for input, then validate).

Q15 **[4 Marks]**

Trace this and give the output: `x=5; IF x>3 THEN x=x*2; PRINT x`.

Sample Answer. Since $5 > 3$ is **true**, x becomes $5 \times 2 = 10$; output **10**. A dry run confirms the branch is taken because the condition holds.

Section C — 6 Mark Questions (Explanation with a diagram or a two-step example; attempt all fifteen)

Q1 **[6 Marks]**

Draw and explain a flowchart to find the largest of two numbers.

Sample Answer. **Start** → input a, b → decision "$a > b$?": **Yes** → print a; **No** → print b → **Stop**. The diamond splits control into two paths that both reach Stop. This is the simplest **selection** flowchart; for equal values the "No" branch prints b (which equals a), so the result is still correct.

Q2 **[6 Marks]**

Draw and explain a flowchart to print numbers from 1 to N using a loop.

Sample Answer. **Start** → input N → set $i = 1$ → decision "$i \le N$?": **Yes** → print i → $i = i + 1$ → back to the decision; **No** → **Stop**. The arrow returning to the decision forms the **loop**. The three loop essentials appear: initialise ($i = 1$), test ($i \le N$), update ($i = i + 1$).

Q3 **[6 Marks]**

Construct the full truth table for A AND (B OR C).

Sample Answer.

A	B	C	$B \vee C$	$A \wedge (B \vee C)$
F	F	F	F	F
F	F	T	T	F
F	T	F	T	F
F	T	T	T	F
T	F	F	F	F
T	F	T	T	T
T	T	F	T	T
T	T	T	T	T

The output is true only when A is true **and** at least one of B, C is true — the last three rows.

Q4 **[6 Marks]**

Build a decision table for a traffic-light rule: "Go if green; Stop if red; Slow if amber."

Sample Answer.

Green?	Amber?	Red?	Action
Y	N	N	Go
N	Y	N	Slow
N	N	Y	Stop

Exactly one condition is true at a time, so the table has three valid rules; any other combination is an invalid signal state. Decision tables make such rule sets explicit and complete.

Q5 **[6 Marks]**

Dry-run this algorithm and give the output using a state table: `a=2; b=3; FOR i=1 TO 3 DO a=a+b; PRINT a`.

Sample Answer.

Step	i	a (with $b = 3$)
init	–	2
1	1	5
2	2	8
3	3	11

b stays 3; a grows by 3 each pass. Output **11**. The state table exposes the running value of a at every step.

Q6 **[6 Marks]**

Explain execution tracing and trace: `n=4; f=1; WHILE n>0 DO f=f*n; n=n-1`. Give the output.

Sample Answer. **Execution tracing** means following each statement and recording variable changes. Trace: n=4, f=1 → f=4, n=3 → f=12, n=2 → f=24, n=1 → f=24, n=0; loop stops. Output **24** (= 4!). Tracing shows this computes the factorial of the initial n.

Q7 **[6 Marks]**

Design a complete set of test cases for a program that classifies a triangle as valid or invalid from three side lengths.

Sample Answer. A triangle is valid if each side is positive and the sum of any two sides exceeds the third. **Test cases**: $(3, 4, 5)$ → valid (normal); $(1, 1, 2)$ → invalid (boundary: $1 + 1 = 2$, not > 2); $(0, 4, 5)$ → invalid (zero side); $(-3, 4, 5)$ → invalid (negative). These cover normal, boundary and invalid inputs, testing every branch of the rule.

Q8 **[6 Marks]**

Explain, with an example, how a dry run detects an off-by-one error in a loop.

Sample Answer. Consider `i=1; WHILE i<5 DO PRINT i; i=i+1` intended to print 1–5. A **dry run** shows it prints $1, 2, 3, 4$ and stops when $i = 5$ (since $5 < 5$ is false) — so 5 is **missing**. The trace reveals the **off-by-one** error; fixing the test to $i \leq 5$ prints all five. This is a bug the compiler cannot catch.

Q9 **[6 Marks]**

Draw and explain a flowchart for checking whether a year is a leap year.

Sample Answer. **Start** → input year → decision "year MOD 400 = 0?": Yes → leap. No → decision "year MOD 100 = 0?": Yes → not-leap; No → decision "year MOD 4 = 0?": Yes → leap, No → not-leap → **Stop**. The nested diamonds implement the leap-year rule; e.g. 2000 (%400) is leap, 1900 (%100 not %400) is not.

Q10 **[6 Marks]**

Compare a flowchart and a decision table for representing multi-condition logic, with an example.

Sample Answer. A **flowchart** shows logic as a path of decisions and is intuitive for *flow*; but with many conditions it becomes a maze of diamonds. A **decision table** lists *all* condition combinations in rows, guaranteeing none is missed — ideal for complex rules like insurance eligibility. For a rule with 3 conditions, the table has $2^3 = 8$ clear rows, whereas the flowchart would nest 3 levels of diamonds. Use flowcharts for flow, decision tables for completeness.

Q11 **[6 Marks]**

Trace this nested-loop pseudocode and give the output: `FOR i=1 TO 2 DO FOR j=1 TO 2 DO PRINT i,j`.

Sample Answer. Outer $i = 1$: inner $j = 1 \to (1,1)$, $j = 2 \to (1,2)$. Outer $i = 2$: $j = 1 \to (2,1)$, $j = 2 \to (2,2)$. Output: **(1,1)(1,2)(2,1)(2,2)**. The inner loop completes fully for each value of the outer loop, giving $2 \times 2 = 4$ pairs.

Q12 **[6 Marks]**

Explain how state tables help in debugging, using a swap-without-temp bug.

Sample Answer. To swap a=5, b=8 a student writes `a=b; b=a`. State table: after `a=b`, a=8, b=8; after `b=a`, a=8, b=8. The table clearly shows both variables became **8** — the original a was lost. Seeing the wrong state pinpoints the bug; the fix is to use a **temporary** variable. State tables turn invisible variable changes into visible evidence.

Q13 **[6 Marks]**

Construct a truth table for NOT(A AND B) and state the law it illustrates.

Sample Answer.

A	B	$A \wedge B$	$\neg(A \wedge B)$
F	F	F	T
F	T	F	T
T	F	F	T
T	T	T	F

This equals $\neg A \vee \neg B$, illustrating **De Morgan's law**: the negation of an AND is the OR of the negations.

Q14 **[6 Marks]**

Write pseudocode with a flowchart description to sum only the even numbers from 1 to N, and dry-run for $N = 5$.

Sample Answer. `sum=0; FOR i=1 TO N DO IF i MOD 2 = 0 THEN sum=sum+i`. Flowchart: a loop containing a decision diamond "i even?" whose Yes branch adds i. Dry run $N = 5$: even values 2 and 4; $sum = 2 \to 6$. Output **6**. The trace confirms only even numbers are added.

Q15 **[6 Marks]**

Explain boundary-value testing and apply it to a rule "ticket free if age < 5 or age > 60".

Sample Answer. Boundary-value testing checks values right at the edges of conditions, where bugs hide. For this rule the edges are 5 and 60: test ages **4** (free), **5** (not free — the rule says < 5), **60** (not free — rule says > 60), **61** (free). These four cases pin down whether the code correctly uses strict vs non-strict comparisons at both boundaries.

Section D — 8 Mark Questions (Full explanation with diagram or complete example; attempt all fifteen)

Q1 **[8 Marks]**

Explain all standard flowchart symbols with their shapes and uses, and draw a flowchart to find the sum and average of three numbers.

Sample Answer. Symbols: oval (terminal, start/stop), parallelogram (I/O), rectangle (process), diamond (decision), arrow (flow), circle (connector). **Flowchart**: Start $\to$ input a, b, c (parallelogram) $\to$ $sum = a + b + c$ (rectangle) $\to$ $avg = sum/3$ (rectangle) $\to$ print sum, avg (parallelogram) $\to$ Stop. This is a pure **sequence** — no decisions or loops — so control flows straight down. Example: a=10, b=20, c=30 $\Rightarrow$ sum=60, avg=20.

Q2 **[8 Marks]**

Explain the three control structures in detail, each with a flowchart description and pseudocode.

Sample Answer. Sequence: steps run in order (stacked boxes); pseudocode `read a; read b; s=a+b`. **Selection**: a diamond branches Yes/No that rejoin; `IF a>b THEN print a ELSE print b`. **Iteration**: a decision loops control back until a condition fails; `i=1; WHILE i≤n DO`

`print i; i=i+1`. All algorithms combine these three; structured programming forbids arbitrary jumps, using only these clean patterns for readable, correct code.

Q3 **[8 Marks]**

Construct the complete truth table for the expression (A AND B) OR (NOT C) and explain when the output is false.

Sample Answer.

A	B	C	$A \wedge B$	$\neg C$	Result
F	F	F	F	T	T
F	F	T	F	F	F
F	T	F	F	T	T
F	T	T	F	F	F
T	F	F	F	T	T
T	F	T	F	F	F
T	T	F	T	T	T
T	T	T	T	F	T

The result is **false** only when $\neg C$ is false (i.e. C true) **and** $A \wedge B$ is false — rows 2, 4, 6. Elsewhere at least one operand of the OR is true.

Q4 **[8 Marks]**

Explain decision tables fully and build one for a discount policy: "10% if member; extra 5% if purchase > 5000". List all rules.

Sample Answer. A **decision table** has condition rows, all combinations as columns/rows, and the resulting actions. **Conditions**: Member? and Purchase>5000?

Member?	Buy>5000?	Discount
Y	Y	15%
Y	N	10%
N	Y	5%
N	N	0%

All $2^2 = 4$ combinations are covered, so no case is forgotten — the key advantage of decision tables for business rules.

Q5 **[8 Marks]**

Perform a complete dry run with a state table for a bubble-sort pass on the list $[3, 1, 2]$ and state the result of one pass.

Sample Answer. One **bubble-sort** pass compares adjacent pairs and swaps if out of order.

Compare	Action	List
(3,1)	$3 > 1$ swap	[1,3,2]
(3,2)	$3 > 2$ swap	[1,2,3]

After one pass the largest element (3) has "bubbled" to the end: **[1,2,3]**. The state table shows each comparison and swap, the essence of tracing a sort.

Q6 **[8 Marks]**

Explain execution tracing and fully trace this algorithm, giving the output: `a=1; b=1; FOR i=3 TO 6 DO c=a+b; PRINT c; a=b; b=c`.

Sample Answer. This generates **Fibonacci** numbers. Trace (start a=1, b=1): i=3 : c=2, print 2, a=1, b=2; i=4 : c=3, print 3, a=2, b=3; i=5 : c=5, print 5, a=3, b=5; i=6 : c=8, print 8, a=5, b=8. Output **2 3 5 8**. Tracing reveals that reassigning a, b each pass slides the pair forward through the sequence.

Q7 **[8 Marks]**

Design a full test-suite (with expected outputs) for a grade calculator that maps marks to A/B/C/F, and justify the choice of each case.

Sample Answer. Rule: A $\geq$ 90, B $\geq$ 80, C $\geq$ 70, else F. **Test suite**: 95 $\rightarrow$ A (normal high); 90 $\rightarrow$ A (boundary); 85 $\rightarrow$ B; 80 $\rightarrow$ B (boundary); 72 $\rightarrow$ C; 70 $\rightarrow$ C (boundary); 69 $\rightarrow$ F; -5 and 105 $\rightarrow$ error (invalid). Boundary cases at 90/80/70 catch >-vs-$\geq$ mistakes; the invalid cases test input validation; normal cases confirm each band. This gives confidence every branch works.

Q8 **[8 Marks]**

Explain the debugging process step by step and illustrate it by finding the bug in `sum=0; FOR i=1 TO n DO sum=i` (meant to total 1..n).

Sample Answer. Debugging steps: reproduce the wrong output, isolate the region, trace the state, form a hypothesis, fix, and re-test. **Trace** for $n = 3$: after the loop, *sum* holds 3 (the last i), not 6. The state table shows *sum* being *overwritten* each pass ($sum = i$) instead of *accumulated*. **Fix**: `sum=sum+i`. Re-test: $1, 3, 6$ — correct. This shows how tracing locates a logic error the compiler accepts.

Q9 **[8 Marks]**

Draw and explain a flowchart with a loop and a decision to count how many numbers in a list of N inputs are positive.

Sample Answer. Start $\rightarrow$ input N $\rightarrow$ $count = 0, i = 1$ $\rightarrow$ decision "$i \leq N$?": No $\rightarrow$ print *count*, Stop. Yes $\rightarrow$ input x $\rightarrow$ decision "$x > 0$?": Yes $\rightarrow$ $count = count + 1$; both branches $\rightarrow$ $i = i + 1$ $\rightarrow$ back to the loop test. The chart nests a **selection** (positive?) inside an **iteration** (over N items), a very common counting pattern.

Q10 **[8 Marks]**

Explain the relationship between truth tables and decision tables, and convert the truth table of A AND B into a decision table with actions "Allow/Deny".

Sample Answer. A **truth table** gives the logical output for input combinations; a **decision table** attaches **actions** to those combinations. For an access rule "Allow only if A AND B":

A	B	Action
T	T	Allow
T	F	Deny
F	T	Deny
F	F	Deny

The logical truth table becomes an actionable decision table by mapping T (both) to **Allow** and every other row to Deny.

Q11 **[8 Marks]**

Trace a linear search with a state table: find 7 in $[4, 7, 1, 7]$; report the first position and discuss finding all positions.

Sample Answer. **Linear search** scans left to right.

i	value	match?
1	4	no
2	7	**yes**

First position **2**; the search stops there. To find **all** positions, remove the early stop and continue: 7 also appears at position 4, so all positions are $\{2, 4\}$. The state table makes the scan and stopping condition explicit.

Q12 **[8 Marks]**

Explain why testing at boundaries and invalid inputs is essential, using a "withdraw amount" banking example, and list five test cases.

Sample Answer. Bugs cluster at **boundaries** (limits) and on **invalid** inputs, which naive testing misses. For "withdraw $\leq$ balance, amount $>$ 0, multiple of 100" with balance 5000: (1) 1000 $\rightarrow$ ok (normal); (2) 5000 $\rightarrow$ ok (boundary equal to balance); (3) 5100 $\rightarrow$ reject (over balance); (4) 150 $\rightarrow$ reject (not multiple of 100); (5) -200 $\rightarrow$ reject (invalid). These cover normal, boundary and invalid paths, catching errors that "happy-path" testing would leave hidden.

Q13 **[8 Marks]**

Given the algorithm to reverse a number (`rev=0; WHILE n>0 DO r=n MOD 10; rev=rev*10+r; n=n/10`), trace it for $n = 123$ with a state table.

Sample Answer.

Step	n	r	rev
init	123	–	0
1	12	3	3
2	1	2	32
3	0	1	321

(Integer division.) When n reaches 0 the loop stops; **rev** $= 321$, the reverse of 123. The trace shows how each digit is peeled off by MOD and built up by $rev \times 10 + r$.

Q14 **[8 Marks]**

Explain structured programming and why the GOTO-free use of the three control structures produces better flowcharts, with an example contrast.

Sample Answer. **Structured programming** builds all logic from **sequence**, **selection** and **iteration**, avoiding arbitrary **GOTO** jumps. A GOTO-laden flowchart has arrows crossing everywhere ("spaghetti"), which is hard to read, trace and debug. The same logic with structured blocks flows cleanly top-to-bottom with well-nested loops and decisions. Example: a menu handled by nested IF/loops is far clearer than one that jumps to distant labels. Structured charts are easier to dry-run, test and maintain.

Q15 **[8 Marks]**

Draw and explain a flowchart, then dry-run, an algorithm that reads N numbers and prints their maximum and minimum. Use the list $[4, 9, 2, 7]$.

Sample Answer. Start → input N and the first number x → set $max = x, min = x, i = 2$ → loop "$i \leq N$?": Yes → read x; "$x > max$?" update max; "$x < min$?" update min; $i = i + 1$ → back to loop; No → print max, min, Stop. **Dry run** $[4, 9, 2, 7]$: start max=min=4; see 9 ⇒ max=9; see 2 ⇒ min=2; see 7 (no change); output **max= 9, min= 2**. The chart nests two **decisions** inside one **loop**, tracking two running values — a common and important pattern.

Section E — 12 Mark Questions

(Comprehensive multi-part answers; attempt all fifteen)

Q1 **[12 Marks]**

(a) List all flowchart symbols with uses. (b) Draw a flowchart to find the largest of three numbers. (c) Dry-run it for $(4, 9, 6)$ with a state table.

Sample Answer. **(a)** Oval (start/stop), parallelogram (I/O), rectangle (process), diamond (decision), arrow (flow), circle (connector). **(b)** Start → input a, b, c → $max = a$ → "$b > max$?" Yes→ $max = b$ → "$c > max$?" Yes→ $max = c$ → print max → Stop. **(c)** State table $(4, 9, 6)$: max=4; b=9 > 4 ⇒ max=9; c=6 > 9? no; output **9**. The running-maximum pattern generalises to any list.

Q2 **[12 Marks]**

(a) Explain the three control structures with flowchart sketches. (b) Write pseudocode using all three to grade N students' marks. (c) Dry-run for marks $[85, 42, 30]$.

Sample Answer. **(a)** Sequence (stacked boxes), selection (diamond branches), iteration (loop back-arrow). **(b)** `FOR k=1 TO N DO READ m; IF m≥40 THEN PRINT "Pass" ELSE PRINT "Fail"`. **(c)** 85 ≥ 40 ⇒ Pass; 42 ≥ 40 ⇒ Pass; 30 ≥ 40? no ⇒ Fail. Output **Pass, Pass, Fail**. The loop (iteration) wraps a decision (selection) over sequential reads.

Q3 **[12 Marks]**

(a) Construct the truth table for (A OR B) AND (NOT A OR C). (b) Identify the rows where it is true. (c) Explain one practical use of truth tables in computing.

Sample Answer. **(a)**

A	B	C	$A \vee B$	$\neg A \vee C$	Result
F	F	F	F	T	F
F	F	T	F	T	F
F	T	F	T	T	T
F	T	T	T	T	T
T	F	F	T	F	F
T	F	T	T	T	T
T	T	F	T	F	F
T	T	T	T	T	T

(b) True in rows 3, 4, 6, 8. **(c)** Truth tables let us **design and simplify digital logic circuits** and verify conditions in programs, ensuring every input combination is handled correctly.

Q4 [12 Marks]

(a) Explain decision tables and their advantage. (b) Build one for a login system with conditions "correct username" and "correct password". (c) Extend it with an "account locked" condition and list all rules.

Sample Answer. **(a)** A **decision table** enumerates all condition combinations with their actions, guaranteeing completeness. **(b)** 2 conditions $\Rightarrow$ 4 rules: only (user Y, pass Y) $\rightarrow$ Login; others $\rightarrow$ Reject. **(c)** Adding "locked" gives $2^3 = 8$ rows; any row with *locked* = Y $\rightarrow$ "Access denied (locked)" regardless of the rest, and only (user Y, pass Y, locked N) $\rightarrow$ Login, all others Reject. The table shows the lock overrides correct credentials — a rule easy to miss without it.

Q5 [12 Marks]

(a) Explain dry running and state tables. (b) Trace `GCD` by subtraction for $(12, 8)$. (c) State the output and one benefit of tracing.

Sample Answer. **(a)** A **dry run** executes code by hand; a **state table** records variables per step. **(b)** Euclid by subtraction: while $a \neq b$, subtract the smaller from the larger.

Step	a	b
init	12	8
1	4	8
2	4	4

(c) When $a = b = 4$, the **GCD is 4**. **Benefit**: tracing verifies the loop terminates and returns the right value before any coding.

Q6 [12 Marks]

(a) Explain the difference between syntax, runtime and logic errors. (b) Give an example of each. (c) Explain which type dry runs and test cases are best at catching.

Sample Answer. **(a)** **Syntax** errors break grammar (caught at compile time); **runtime** errors crash during execution (e.g. divide by zero); **logic** errors run fine but give wrong results. **(b)** Syntax: missing semicolon; runtime: `x/0`; logic: using + instead of −. **(c)** **Dry runs and test cases** are best at catching **logic errors**, because those produce wrong output without any crash

or compiler complaint — only checking values against expected results reveals them.

Q7 **[12 Marks]**

(a) Draw a flowchart for a number-guessing loop (guess vs secret). (b) Explain the loop and decisions. (c) Dry-run with secret = 7 and guesses 5, 9, 7.

Sample Answer. **(a)** Start → set secret → input guess → "guess = secret?" Yes → print "Correct", Stop; No → "guess < secret?" Yes → "Too low" / No → "Too high" → back to input. **(b)** A **loop** repeats until the guess matches; a **nested decision** gives the low/high hint. **(c)** guess 5 → Too low; 9 → Too high; 7 → **Correct**, stop. The trace confirms the hints and the exit condition.

Q8 **[12 Marks]**

(a) Explain iteration types (FOR, WHILE, DO–WHILE). (b) Give pseudocode of each printing 1..3. (c) State which runs at least once and dry-run the DO–WHILE.

Sample Answer. **(a)** **FOR**: count-controlled; **WHILE**: pre-test; **DO–WHILE**: post-test. **(b)** FOR: `FOR i=1 TO 3 PRINT i`; WHILE: `i=1; WHILE i≤3 DO PRINT i; i=i+1`; DO–WHILE: `i=1; DO PRINT i; i=i+1 WHILE i≤3`. **(c)** **DO–WHILE** always runs at least once. Dry run: print 1 ($i \to 2$), print 2 ($i \to 3$), print 3 ($i \to 4$), $4 \leq 3$ false, stop ⇒ **1 2 3**.

Q9 **[12 Marks]**

(a) Build a decision table for a fee rule: hosteller pays extra; scholarship waives tuition. (b) List all rules. (c) Convert one rule into an IF–statement.

Sample Answer. **(a)** Conditions: Hosteller? and Scholarship?

Hosteller?	Scholarship?	Fee
Y	Y	Hostel only (tuition waived)
Y	N	Tuition + hostel
N	Y	Zero (tuition waived)
N	N	Tuition only

(b) Four rules as above cover all 2^2 cases. **(c)** Rule 2: `IF hosteller AND NOT scholarship THEN fee = tuition + hostel`.

Q10 **[12 Marks]**

(a) Explain test-case design (normal, boundary, invalid, and combinations). (b) Design a suite for "BMI category". (c) Explain how tracing complements testing.

Sample Answer. **(a)** **Normal** cases hit typical values; **boundary** cases hit band edges; **invalid** cases hit impossible inputs; **combinations** mix conditions. **(b)** BMI: 18.4 → underweight, 18.5 → normal (boundary), 24.9/25.0 → normal/overweight (boundary), 30 → obese, height = 0 → error (invalid). **(c)** **Tracing** shows *why* a case fails by exposing intermediate values, while **testing** shows *that* it fails; used together, testing flags the bug and tracing locates it.

Q11 **[12 Marks]**

(a) Trace selection-sort's first pass on $[5, 3, 8, 1]$ with a state table. (b) State the list after the pass. (c) Explain how many passes the full sort needs.

Sample Answer. **(a)** First pass finds the minimum and swaps it to the front.

Scan	min-so-far	note
5	5	start
3	3	smaller
8	3	keep
1	1	smaller

Swap min (1) with the first element. **(b)** List becomes **[1,3,8,5]**. **(c)** For $n = 4$ elements, selection sort needs $n - 1 = 3$ passes, each placing one more element in its final position.

Q12 **[12 Marks]**

(a) Explain how flowcharts, decision tables and dry runs together support the software process. (b) Illustrate with a small "ATM PIN check (3 tries)" example. (c) State the benefit of each technique.

Sample Answer. **(a)** **Flowcharts** design the control flow, **decision tables** enumerate the rules, and **dry runs** verify behaviour before coding. **(b)** For PIN check: a flowchart loops up to 3 times (decision "PIN correct?" / counter); a decision table maps (correct?, tries-left?) to Allow/Retry/Block; a dry run with a wrong-wrong-right sequence confirms access on the third try, and wrong×3 blocks the card. **(c)** Flowchart → clarity of flow; decision table → completeness of rules; dry run → early error detection.

Q13 **[12 Marks]**

(a) Construct truth tables for AND, OR, NOT, XOR. (b) Explain XOR's special property. (c) Give one computing use of XOR.

Sample Answer. **(a)** AND true only for (T,T); OR false only for (F,F); NOT inverts; **XOR** is true when inputs *differ*:

A	B	$A \oplus B$
F	F	F
F	T	T
T	F	T
T	T	F

(b) XOR outputs true **iff the inputs are unequal**, so $A \oplus A = 0$ and $A \oplus 0 = A$. **(c)** XOR is used in **parity checks**, simple encryption and computing checksums, because applying the same key twice restores the original.

Q14 **[12 Marks]**

(a) Explain complete execution tracing. (b) Trace `power`: `res=1; FOR i=1 TO e DO res=res*b` for $b = 2, e = 4$. (c) Verify the result and state one use of tracing in AI programming.

Sample Answer. **(a)** **Execution tracing** records every variable after each statement to verify logic. **(b)** State table b=2:

i	res
init	1
1	2
2	4
3	8
4	16

(c) Output **16** $= 2^4$. ✓In AI programming, tracing helps debug data pipelines and loops (e.g. checking how a weight or counter updates each iteration), so the same skill scales from basic algorithms to ML code.

Q15 **[12 Marks]**

(a) Summarise Unit 3: flowcharts, control structures, logic tables, tracing, testing. (b) Explain how these ensure a program is correct. (c) Link these skills to debugging AI/ML programs.

Sample Answer. **(a)** We learned **flowcharts** (visual logic), **control structures** (sequence/selection/iteration), **decision and truth tables** (rule logic), **dry runs/state tables** (tracing), and **test-case design/debugging**. **(b)** Together they let us design the logic, enumerate all cases, trace behaviour and test edges — catching **logic errors** the compiler cannot, so the program does what is intended. **(c)** AI/ML code is full of loops, conditions and data transformations; the same tracing and testing discipline is exactly how engineers debug training loops, data filters and model logic — making Unit 3 a foundation for reliable AI development.

4 Unit 4 — Introduction to Artificial Intelligence

Syllabus. A. Definition, scope, concepts and history of AI. **B.** Goals, applications and domains; AI learning pathway; intelligent agents and rational behaviour. **C.** AI vs traditional problem solving; philosophical and practical aspects; ethics. *(CO4, K1–K4)*

Section A — 2 Mark Questions

(Very short answers; attempt all fifteen)

Q1 **[2 Marks]**

Define Artificial Intelligence.

Sample Answer. **Artificial Intelligence (AI)** is the branch of computer science that builds machines and programs able to perform tasks that normally require human intelligence, such as reasoning, learning and perception.

Q2 **[2 Marks]**

Who coined the term "Artificial Intelligence" and in which year?

Sample Answer. **John McCarthy** coined the term in **1956**, at the Dartmouth Conference.

Q3 **[2 Marks]**

What is the Turing Test?

Sample Answer. The **Turing Test** (Alan Turing, 1950) judges a machine as intelligent if a human cannot reliably tell its responses from a human's in a text conversation.

Q4 **[2 Marks]**

Name the four approaches to defining AI.

Sample Answer. Thinking humanly, **thinking rationally**, acting humanly, and **acting rationally**.

Q5 **[2 Marks]**

What is a rational agent?

Sample Answer. A **rational agent** is one that acts to achieve the best expected outcome (maximise its performance measure) given what it perceives and knows.

Q6 **[2 Marks]**

What does PEAS stand for?

Sample Answer. **P**erformance measure, **E**nvironment, **A**ctuators, **S**ensors — the framework used to specify an agent's task.

Q7 **[2 Marks]**

Differentiate strong AI and weak AI in one line.

Sample Answer. **Weak (narrow) AI** is built for one specific task; **strong (general) AI** would match human intelligence across any task (still hypothetical).

Q8 **[2 Marks]**

Name any two sub-domains of AI.

Sample Answer. Any two of: **Machine Learning**, **Natural Language Processing**, **Computer Vision**, **Robotics**, **Expert Systems**.

Q9 **[2 Marks]**

What is Machine Learning in relation to AI?

Sample Answer. **Machine Learning** is a subset of AI in which systems learn patterns from data and improve with experience instead of being explicitly programmed.

Q10 **[2 Marks]**

What is an intelligent agent?

Sample Answer. An **intelligent agent** is anything that perceives its environment through sensors and acts upon it through actuators to achieve its goals.

Q11 **[2 Marks]**

Name two real-world applications of AI in daily life.

Sample Answer. Any two of: **voice assistants** (Siri/Alexa), **recommendations** (Netflix/YouTube), **navigation** (Google Maps), **spam filtering**, **face unlock**.

Q12 **[2 Marks]**

What is the Chinese Room argument about?

Sample Answer. The **Chinese Room** (John Searle) argues that a system can manipulate symbols and appear to understand language without any **real understanding**, challenging strong AI.

Q13 **[2 Marks]**

What is an AI winter?

Sample Answer. An **AI winter** is a period of reduced funding and interest in AI after expectations were not met; notable winters occurred in the mid-1970s and late 1980s.

Q14 **[2 Marks]**

State one goal of Artificial Intelligence.

Sample Answer. To build systems that can **reason, learn, perceive and act** so as to solve problems and assist humans (any one such goal).

Q15 **[2 Marks]**

Name one AI initiative or application relevant to India.

Sample Answer. Any one of: **Digital India** AI programmes, AI in agriculture (crop advice), AI in healthcare (diagnosis in rural areas), or vernacular-language chatbots.

Section B — 4 Mark Questions

(Short answers with an example; attempt all fifteen)

Q1 **[4 Marks]**

Explain the four approaches to AI with one line on each.

Sample Answer. **Acting humanly** — pass the Turing Test (behave like a human). **Thinking humanly** — model human thought (cognitive science). **Thinking rationally** — reason by logic laws. **Acting rationally** — the *rational-agent* view, doing the right thing to maximise goals. Modern AI mainly follows the **acting rationally** approach as it is measurable and general.

Q2 **[4 Marks]**

Explain the Turing Test and state one common objection to it.

Sample Answer. In the **Turing Test**, a human judge chats (by text) with a hidden human and a hidden machine; if the judge cannot reliably tell them apart, the machine is deemed "intelligent". **Objection**: passing it may show clever imitation, not real understanding — the basis of Searle's **Chinese Room**. It tests behaviour, not consciousness.

Q3 **[4 Marks]**

Describe the four historical phases of AI development with approximate years.

Sample Answer. **Birth** (1950–56): Turing Test, Dartmouth Conference. **Early enthusiasm** (1956–74): logic and search programs. **AI winters** (1974–80, 1987–93): funding cuts after over-promising. **Modern boom** (1997–present): Deep Blue (1997), deep learning (2012), AlphaGo (2016), ChatGPT (2022). The field has moved through hype, disappointment and renewed

success driven by data and computing power.

Q4 **[4 Marks]**

Explain the PEAS framework using a self-driving car as an example.

Sample Answer. **PEAS** specifies an agent's task: **Performance** — safety, speed, comfort, legality; **Environment** — roads, traffic, pedestrians, weather; **Actuators** — steering, accelerator, brake; **Sensors** — cameras, LiDAR, GPS, radar. Defining PEAS clarifies exactly what the agent must optimise and with what inputs and outputs.

Q5 **[4 Marks]**

Differentiate strong AI and weak AI with examples.

Sample Answer. **Weak (narrow) AI** performs one task well but has no general intelligence — e.g. a chess engine or a spam filter; all AI today is of this kind. **Strong (general) AI** would understand and learn *any* task like a human, and remains hypothetical. The key difference is **scope**: narrow vs human-level general ability.

Q6 **[4 Marks]**

List and briefly describe any four sub-domains of AI.

Sample Answer. **Machine Learning** — learning from data; **Natural Language Processing** — understanding and generating language; **Computer Vision** — interpreting images and video; **Robotics** — intelligent physical machines. (Also expert systems, planning.) Together these sub-domains give machines the abilities to learn, communicate, see and act.

Q7 **[4 Marks]**

Explain how AI differs from traditional (conventional) programming.

Sample Answer. In **traditional programming**, a human writes explicit rules: input + rules → output. In **AI/ML**, the machine *learns* the rules from examples: input + output → rules (a model). Traditional code is best when rules are clear (payroll); AI is best when rules are hard to state (face recognition).

Q8 **[4 Marks]**

State four real-world applications of AI across different domains.

Sample Answer. **Healthcare** — disease detection from scans; **Finance** — fraud detection; **Transport** — self-driving and traffic prediction; **Education** — personalised tutoring. (Also agriculture, retail, entertainment.) AI adds value wherever there is data and a task too complex for fixed rules.

Q9 **[4 Marks]**

Explain the concept of an intelligent agent with a labelled description of its parts.

Sample Answer. An **intelligent agent** *perceives* its environment through **sensors**, *decides* using an internal **agent program**, and *acts* through **actuators** (Percept → Agent → Action, in a loop with the environment). Example: a robot vacuum senses dirt/walls, decides where to move, and drives its wheels and brush.

Q10 **[4 Marks]**

Name and briefly describe two types of intelligent agents.

Sample Answer. **Simple reflex agent** — acts only on the current percept via condition–action rules (e.g. "if dirty then clean"). **Goal-based agent** — considers future outcomes to reach a goal (e.g. a route planner). (Others: model-based, utility-based, learning agents.) More advanced agents keep internal state and plan ahead.

Q11 **[4 Marks]**

Explain the Chinese Room argument and what it tries to prove.

Sample Answer. In the **Chinese Room**, a person who knows no Chinese follows a rule book to answer Chinese notes correctly, appearing fluent without *understanding* anything. Searle argues that a computer likewise manipulates symbols by rules without genuine **understanding**, so passing the Turing Test does not prove **strong AI** or real comprehension. It targets the claim that syntax alone yields semantics.

Q12 **[4 Marks]**

Discuss two ethical concerns in AI with an example of each.

Sample Answer. **Bias/fairness**: a hiring model trained on biased data may unfairly reject qualified candidates. **Privacy**: face-recognition surveillance can track people without consent. (Also job loss, accountability, misinformation.) Responsible AI requires fair data, transparency and human oversight to prevent harm.

Q13 **[4 Marks]**

Explain the AI learning pathway a student should follow.

Sample Answer. Start with **maths** (algebra, probability, statistics) and **programming** (Python); learn **data handling**; then **machine learning** basics; then specialise in **deep learning**, NLP or vision; finally build **projects**. Strong fundamentals in computing (Units 1–3 of this course) underpin the whole pathway.

Q14 **[4 Marks]**

Give four goals of AI as a field.

Sample Answer. To make machines that can **reason** (draw conclusions), **learn** (improve from data), **perceive** (see, hear, read) and **act** (plan and move) intelligently. The broader aim is to **assist and augment** human capabilities across science, industry and daily life.

Q15 **[4 Marks]**

Explain, with examples, when AI is advantageous over traditional methods and when it is not.

Sample Answer. AI is **advantageous** when the rules are unknown or too complex and data is plentiful — e.g. speech recognition, image tagging, recommendations. It is **not** ideal when rules are clear and exact answers are required — e.g. computing tax or sorting numbers, where traditional code is simpler, faster and fully predictable. Choosing the right approach depends on the problem.

Section C — 6 Mark Questions

(Explanation with a small diagram or structured points; attempt all fifteen)

Q1 **[6 Marks]**

Explain the four approaches to AI in detail with the two dimensions (human vs rational, thought vs behaviour).

Sample Answer. The four approaches sit on two axes: *human-like vs rational*, and *thinking vs acting*. **Thinking humanly** (cognitive modelling) and **acting humanly** (Turing Test) compare AI to *humans*; **thinking rationally** (logic) and **acting rationally** (rational agents) compare it to an *ideal* standard. Humans are not always rational, so mimicking humans and being optimal differ. Modern AI favours **acting rationally**: it is well-defined, measurable and general, and it frames AI as building agents that do the right thing.

Q2 **[6 Marks]**

Draw and explain the agent–environment interaction diagram.

Sample Answer. (Diagram: an **Agent** box and an **Environment** box; the environment sends **percepts** to the agent via sensors, and the agent sends **actions** back via actuators, forming a loop.) The agent's **program** maps percept histories to actions. Example: a thermostat senses temperature (percept) and switches heating (action). This perceive–decide–act loop is the core abstraction of all AI agents.

Q3 **[6 Marks]**

Explain the historical milestones of AI from 1950 to 2022 with at least six events.

Sample Answer. **1950** Turing Test; **1956** Dartmouth Conference coins "AI" (McCarthy); **1966** ELIZA chatbot; **1997** Deep Blue beats Kasparov at chess; **2011** IBM Watson wins Jeopardy!; **2012** deep learning breakthrough (ImageNet); **2016** AlphaGo beats a Go champion; **2022** ChatGPT popularises generative AI. The arc moves from symbolic logic to data-driven deep learning, punctuated by two AI winters.

Q4 **[6 Marks]**

Specify the PEAS description for two different agents (a medical-diagnosis system and a chess program).

Sample Answer. **Medical-diagnosis**: P — correct diagnosis; E — patient, symptoms, tests; A — report/recommendation on screen; S — entered symptoms and test data. **Chess program**: P — win/draw; E — the board and opponent; A — make a move; S — read the board state. PEAS makes each agent's task, inputs and outputs explicit and comparable.

Q5 **[6 Marks]**

Compare AI-based and traditional problem solving on at least five points in a table.

Sample Answer.

Aspect	Traditional	AI-based
Rules	Hand-coded by humans	Learned from data
Data need	Low	High
Adaptability	Fixed	Improves with data
Best for	Clear, exact tasks	Complex, fuzzy tasks
Output	Exact, predictable	Often probabilistic

Traditional methods suit well-defined problems; AI suits perception and pattern tasks where rules cannot be written by hand.

Q6 **[6 Marks]**

Explain the five types of intelligent agents in increasing sophistication.

Sample Answer. **Simple reflex** — acts on the current percept via if–then rules. **Model-based reflex** — keeps internal state to handle partial views. **Goal-based** — chooses actions that reach a goal. **Utility-based** — picks actions maximising a utility (quality) measure among goals. **Learning agent** — improves its behaviour from experience. Each level adds memory, foresight, preference or learning, moving from reactive to truly adaptive behaviour.

Q7 **[6 Marks]**

Explain the concepts of environment types (fully vs partially observable, deterministic vs stochastic, static vs dynamic).

Sample Answer. **Fully observable**: sensors give all needed state (chess) vs **partially observable**: hidden information (poker, driving). **Deterministic**: the next state is fixed by the action (a puzzle) vs **stochastic**: chance is involved (dice, real roads). **Static**: the world waits for the agent vs **dynamic**: it changes while the agent thinks (traffic). Harder environments (partially observable, stochastic, dynamic) demand more sophisticated agents.

Q8 **[6 Marks]**

Explain strong AI, weak AI, and super AI with examples and current status.

Sample Answer. **Weak/narrow AI**: single-task systems — chatbots, recommenders — this is all AI today. **Strong/general AI (AGI)**: human-level ability across *any* task — still research, not achieved. **Super AI**: hypothetical intelligence far beyond humans — purely speculative. The progression is narrow → general → super, and we are firmly at the **narrow** stage.

Q9 **[6 Marks]**

Discuss five domains where AI is applied in India, with the technique used in each.

Sample Answer. **Agriculture** — ML crop and pest advice; **Healthcare** — image-based diagnosis for rural clinics; **Finance** — fraud detection and credit scoring; **Language** — NLP chatbots in Hindi and regional languages; **Governance** — AI in traffic and public services (Digital India). Each addresses scale and diversity challenges where fixed rules fall short.

Q10 **[6 Marks]**

Explain the Chinese Room argument in detail and give one counter-argument.

Sample Answer. Searle imagines a person in a room using a rule book to reply to Chinese messages perfectly, without understanding Chinese; he concludes computers similarly lack real **understanding**, only symbol manipulation. **Counter (systems reply)**: although the person doesn't understand, the *whole system* (person + rules + memory) may be said to understand, just as no single neuron understands but the brain does. The debate concerns whether behaviour equals understanding.

Q11 **[6 Marks]**

Explain the goals and scope of AI, distinguishing short-term and long-term aims.

Sample Answer. **Short-term goals**: build useful narrow systems — better vision, speech, translation, recommendation and automation. **Long-term goals**: achieve general, adaptable intelligence and safe, beneficial autonomous systems. **Scope** spans reasoning, learning, perception, language and action across nearly every industry. Balancing capability with **safety and ethics** is central to the field's direction.

Q12 **[6 Marks]**

Explain the practical challenges and limitations of current AI.

Sample Answer. **Data dependence** — needs large, good-quality data; **bias** — unfair data gives unfair results; **black-box** — deep models are hard to explain; **brittleness** — can fail on inputs unlike its training; **cost** — heavy computing needs; **no common sense** — lacks true understanding. These limits mean humans must supervise AI and use it responsibly, especially in high-stakes settings.

Q13 **[6 Marks]**

With a diagram, explain where Machine Learning and Deep Learning sit within AI.

Sample Answer. (Diagram: three nested circles — outer **AI**, inside it **Machine Learning**, inside that **Deep Learning**.) **AI** is the broad goal of intelligent behaviour; **ML** is the subset that learns from data; **Deep Learning** is the subset of ML using multi-layer neural networks. So all deep learning is ML, and all ML is AI, but not vice versa — a useful map of the field.

Q14 **[6 Marks]**

Explain rationality and bounded rationality with an example.

Sample Answer. **Rationality** means choosing the action that maximises the expected performance measure given the agent's knowledge. In practice, agents have limited time, data and computing, so they use **bounded rationality** — making the best decision *possible* within those limits. Example: a chess engine cannot search every move, so it searches a few levels deep and uses heuristics — good enough, not perfect. Real AI is boundedly rational.

Q15 **[6 Marks]**

Explain how AI, ML and traditional programming relate, using "recognise handwritten digits" as the example.

Sample Answer. **Traditional**: a programmer would try to hand-write rules for every digit shape — nearly impossible given handwriting variety. **AI/ML**: instead we show the machine thousands of labelled digit images and it *learns* the patterns, producing a model that classifies new digits. This task shows why AI is needed: the rules are too complex to state, but examples are plentiful — a classic case for machine learning over conventional code.

Section D — 8 Mark Questions

(Full explanation with diagram or structured discussion; attempt all fifteen)

Q1 **[8 Marks]**

Explain in detail the definition, scope and importance of AI, and discuss why it is called an interdisciplinary field.

Sample Answer. **AI** builds systems that reason, learn, perceive and act intelligently. Its **scope** spans machine learning, vision, language, robotics and expert systems, and its **importance** lies in solving problems too complex for fixed rules — from diagnosis to translation. AI is **interdisciplinary** because it draws on **computer science** (algorithms), **mathematics** (probability, linear algebra, optimisation), **statistics** (learning from data), **neuroscience** (neural inspiration), **linguistics** (NLP) and **philosophy** (mind, ethics). This breadth is why AI both influences and borrows from many fields, and why a strong foundation across subjects helps an AI engineer.

Q2 **[8 Marks]**

Discuss the complete history of AI, including the two AI winters, with causes and recovery.

Sample Answer. **Beginnings**: Turing's 1950 test; the 1956 Dartmouth Conference names the field. **Early optimism** (1956–74): logic, search and early neural nets, with grand promises. **First winter** (1974–80): progress stalled, funding was cut when systems could not scale.

Expert-systems boom (1980s) then the **second winter** (1987–93) as expensive expert systems disappointed. **Recovery** (1997–now): Deep Blue (1997), cheap data and GPUs, deep learning (2012), AlphaGo (2016) and ChatGPT (2022). The pattern — hype, disappointment, data-driven revival — teaches realistic expectations about AI.

Q3 **[8 Marks]**

Explain the rational-agent approach in depth: rationality, performance measure, and the PEAS specification, with a full example.

Sample Answer. The **rational-agent** view defines AI as building agents that *do the right thing*. An agent is **rational** if it selects actions expected to maximise its **performance measure**, given its percepts and knowledge. **PEAS** specifies the task: Performance, Environment, Actuators, Sensors. **Example (automated taxi)**: P — safety, legal, fast, comfortable, profit; E — roads, traffic, pedestrians, weather; A — steering, throttle, brake, horn, display; S — cameras, LiDAR, GPS, speedometer. Specifying PEAS turns a vague goal ("drive well") into a precise engineering target the agent can optimise.

Q4 **[8 Marks]**

Compare and contrast AI-based and traditional problem-solving approaches in detail, with two examples where each is preferable.

Sample Answer. **Traditional** programming encodes human-written rules and gives exact, predictable output; it suits problems with clear logic and modest data. **AI** learns rules from data, adapts and handles fuzzy patterns, but needs much data and may be probabilistic and hard to explain. **Traditional is better** for computing salaries or sorting a list — clear rules, exact answers. **AI is better** for recognising faces or translating language — rules too complex to write, but examples abundant. A good engineer diagnoses the problem type first, then picks the approach; sometimes both are combined.

Q5 **[8 Marks]**

Explain the five types of agents with a diagram-style description and one real example of each.

Sample Answer. **Simple reflex** (thermostat) — condition–action rules on the current percept. **Model-based reflex** (robot vacuum with a map) — maintains internal state for a partly hidden world. **Goal-based** (route planner) — searches actions that achieve a goal. **Utility-based** (self-driving car balancing speed, safety, comfort) — maximises a utility across competing goals. **Learning agent** (recommendation system) — improves from feedback. Structurally, each adds a capability — state, goals, utility, learning — over the previous, moving from reactive to adaptive.

Q6 **[8 Marks]**

Discuss the ethical and societal implications of AI in detail, covering bias, privacy, jobs and accountability, with mitigations.

Sample Answer. **Bias**: models trained on skewed data can discriminate — mitigate with fair, representative data and audits. **Privacy**: AI surveillance and data collection threaten personal freedom — mitigate with consent, anonymisation and regulation. **Jobs**: automation

may displace some roles while creating others — mitigate with reskilling. **Accountability**: when an AI errs, who is responsible? — mitigate with transparency, human oversight and clear liability. Responsible AI keeps humans in control and treats fairness, privacy and safety as design requirements, not afterthoughts.

Q7 **[8 Marks]**

Explain the domains/sub-fields of AI in detail, with the kind of problem each solves and one application.

Sample Answer. **Machine Learning** — learns from data; e.g. credit scoring. **Deep Learning** — multi-layer networks for perception; e.g. image recognition. **NLP** — language understanding/generation; e.g. chatbots and translation. **Computer Vision** — interpreting images/video; e.g. medical imaging. **Robotics** — intelligent physical action; e.g. warehouse robots. **Expert Systems** — rule-based reasoning; e.g. diagnosis aids. Together these give machines the abilities to learn, communicate, see, move and reason.

Q8 **[8 Marks]**

Explain the Turing Test in depth, including its variants and criticisms, and give your reasoned view on its usefulness.

Sample Answer. The **Turing Test** judges intelligence by indistinguishable *behaviour* in text conversation. Variants include the **Total Turing Test** (adding vision and robotics) and modern chatbot evaluations. **Criticisms**: it tests imitation, not understanding (Chinese Room); it rewards deception; and it ignores non-human forms of intelligence. **View**: it was historically important for framing the question operationally, but it is neither necessary nor sufficient for true intelligence today; task-based and capability benchmarks are now more informative. It remains a valuable thought experiment about mind and behaviour.

Q9 **[8 Marks]**

Explain the concept of learning agents with a labelled four-component structure and an example.

Sample Answer. A **learning agent** has four parts: the **performance element** (chooses actions), the **critic** (judges results against a standard), the **learning element** (improves the performance element using the critic's feedback), and the **problem generator** (suggests exploratory actions). **Example**: a game-playing agent plays (performance), sees whether it won (critic), adjusts its strategy (learning), and tries new moves to learn more (problem generator). This structure lets the agent get better with experience — the essence of ML inside the agent framework.

Q10 **[8 Marks]**

Discuss AI applications across five sectors in depth, naming the AI technique and the benefit in each.

Sample Answer. **Healthcare** — computer vision detects tumours in scans, aiding early diagnosis. **Finance** — ML flags fraudulent transactions in real time. **Transport** — vision + reinforcement learning enable driver assistance and route optimisation. **Agriculture** — ML

predicts crop disease and yield from images and weather. **Education** — NLP powers personalised tutoring and auto-grading. In each, AI adds value by handling scale and complexity beyond hand-coded rules, though human oversight remains essential for trust and safety.

Q11 **[8 Marks]**

Explain why AI is considered the "new electricity" and discuss its transformative and disruptive effects with examples.

Sample Answer. Like **electricity**, AI is a **general-purpose technology** that can improve almost every sector. **Transformative**: faster diagnosis, smarter agriculture, accessible translation, and automation of dull tasks raise productivity. **Disruptive**: some jobs change or vanish, decision-making becomes opaque, and misuse (deepfakes, surveillance) poses risks. Just as societies built safety standards and grids for electricity, they must build **governance, skills and ethics** for AI so its benefits are widely and safely shared.

Q12 **[8 Marks]**

Explain the philosophical questions AI raises about mind, consciousness and understanding, referencing key arguments.

Sample Answer. AI forces old questions into practice: *Can a machine think?* The **Turing Test** answers behaviourally, while the **Chinese Room** insists symbol manipulation is not **understanding**. *Can a machine be conscious?* We can measure behaviour but not inner experience, so consciousness remains unresolved. *Strong vs weak AI*: does the system merely simulate mind (weak) or truly have one (strong)? These debates matter practically for rights, responsibility and trust as AI grows more capable, and they keep philosophy central to computer science.

Q13 **[8 Marks]**

Trace the AI learning pathway in detail and map each stage to skills and to units of this course.

Sample Answer. **Stage 1 — Foundations**: computing basics and number systems (Unit 1), problem solving and algorithms (Unit 2), logic and tracing (Unit 3). **Stage 2 — Programming & maths**: Python, probability, linear algebra, statistics. **Stage 3 — Core AI**: intelligent agents and search (Unit 4). **Stage 4 — Machine Learning**: paradigms and algorithms (Unit 5). **Stage 5 — Specialisation & projects**: deep learning, NLP, vision, and building real applications. The course deliberately builds this ladder, so each unit is a rung toward becoming an AI/ML engineer.

Q14 **[8 Marks]**

Explain, with examples, the difference between narrow AI achievements and the challenges of achieving general AI.

Sample Answer. **Narrow AI** already beats humans at specific tasks — chess (Deep Blue), Go (AlphaGo), image labelling and language generation (ChatGPT) — because each is a bounded, data-rich problem. **General AI** must transfer knowledge across *any* task, use common sense, learn from little data, and set its own goals — abilities humans have but machines lack. Chal-

lenges include **common-sense reasoning**, **robustness**, **transfer learning** and **grounding** meaning. Thus stunning narrow successes do not add up to general intelligence, which remains an open research frontier.

Q15 **[8 Marks]**

Explain how a search-based agent solves a problem, using a simple route-finding example, and state why heuristics are used.

Sample Answer. A **search-based agent** represents the problem as **states** (e.g. cities), an **initial state**, **actions** (roads to adjacent cities), a **goal test**, and a **path cost**; it then explores possible paths to reach the goal cheaply. **Example**: to drive from city A to city G, the agent expands routes, keeping track of the cost so far, until it reaches G by the shortest total distance. Because the number of paths grows explosively, a **heuristic** (an informed estimate of the remaining distance, e.g. straight-line distance to G) guides the search toward the goal, so it finds a good route without examining every possibility. This "formulate, search, execute" pattern is a classic AI problem-solving method.

Section E — 12 Mark Questions

(Comprehensive multi-part answers; attempt all fifteen)

Q1 **[12 Marks]**

(a) Define AI and explain the four approaches. (b) Explain the two dimensions that organise them. (c) State which approach modern AI follows and why.

Sample Answer. **(a)** **AI** builds machines that reason, learn, perceive and act. The four approaches: acting humanly (Turing Test), thinking humanly (cognitive modelling), thinking rationally (logic), acting rationally (rational agents). **(b)** Two axes: *human-like vs rational (ideal)* and *thinking vs acting*. **(c)** Modern AI follows **acting rationally**: it is measurable, general and does not require copying imperfect human thought — an agent simply acts to maximise its goals.

Q2 **[12 Marks]**

(a) Explain the history of AI with six milestones. (b) Explain the causes of the AI winters. (c) Explain what caused the current AI boom.

Sample Answer. **(a)** 1950 Turing Test; 1956 Dartmouth (McCarthy); 1997 Deep Blue; 2011 Watson; 2016 AlphaGo; 2022 ChatGPT. **(b)** The **winters** (mid-1970s, late 1980s) came from **over-promising**: limited computing, small data and disappointing expert systems led to funding cuts. **(c)** The **boom** rests on three pillars: **big data** (internet, sensors), **computing power** (GPUs) and **better algorithms** (deep learning since 2012). Together they turned decades-old ideas into practical systems.

Q3 **[12 Marks]**

(a) Explain the rational agent and PEAS. (b) Give PEAS for a vacuum-cleaner robot and a spam filter. (c) Discuss what makes each environment easy or hard.

Sample Answer. **(a)** A **rational agent** maximises its performance measure given percepts; **PEAS** = Performance, Environment, Actuators, Sensors. **(b)** *Vacuum robot*: P — clean floor area; E — rooms, dirt, obstacles; A — wheels, brush, vacuum; S — bump/dirt/cliff sensors. *Spam filter*: P — correct spam/ham labels; E — incoming email; A — move to Spam/Inbox; S — email text and metadata. **(c)** The vacuum's world is **partially observable, dynamic** (harder); the spam filter's is largely **static and fully observable** per email (easier), though spammers adapt over time.

Q4 **[12 Marks]**

(a) Explain the types of intelligent agents. (b) Give a real example of each. (c) Explain how a learning agent improves over time.

Sample Answer. **(a)** Simple reflex, model-based reflex, goal-based, utility-based, learning agents — increasing in memory, foresight, preference and adaptivity. **(b)** Thermostat; mapped vacuum; GPS route planner; self-driving car balancing objectives; recommendation system. **(c)** A **learning agent** uses a **critic** to judge outcomes against a standard and a **learning element** to update its decision-making, while a **problem generator** explores new actions; over many experiences it steadily improves — the agent-level view of machine learning.

Q5 **[12 Marks]**

(a) Compare AI and traditional programming on six points. (b) Give an example best solved by each. (c) Explain a hybrid system that uses both.

Sample Answer. **(a)** Rules (coded vs learned), data need (low vs high), adaptability (fixed vs improving), transparency (clear vs often opaque), best-for (exact vs fuzzy tasks), output (exact vs probabilistic). **(b)** Traditional: payroll calculation; AI: handwriting recognition. **(c)** A **hybrid** email system uses *traditional rules* for fixed policies (block a blacklisted address) and an *ML model* for fuzzy spam detection — combining predictable control with learned pattern recognition, which is common in real products.

Q6 **[12 Marks]**

(a) Explain strong vs weak AI. (b) Explain the Turing Test and the Chinese Room. (c) Give your reasoned position on whether passing the Turing Test proves understanding.

Sample Answer. **(a)** **Weak AI** solves specific tasks (all AI today); **strong AI** would have general, human-level understanding (hypothetical). **(b)** The **Turing Test** equates intelligence with indistinguishable behaviour; the **Chinese Room** replies that rule-based symbol manipulation lacks real **understanding**. **(c)** A reasoned view: passing the Turing Test demonstrates convincing *behaviour* but not necessarily *understanding* or consciousness; behaviour is strong evidence yet not proof, so the test is a useful milestone, not a definition of mind. (Alternative views are acceptable if well argued.)

Q7 **[12 Marks]**

(a) Explain the goals and scope of AI. (b) Describe five sub-domains. (c) Explain how these sub-domains combine in a self-driving car.

Sample Answer. **(a)** **Goals**: reason, learn, perceive, act; **scope**: nearly every industry. **(b)** ML, computer vision, NLP, robotics, planning/expert systems. **(c)** A **self-driving car** combines **computer vision** (detect lanes, signs, pedestrians), **ML/deep learning** (predict behaviour), **planning** (choose a safe path), **robotics/control** (steer, brake) and sometimes **NLP** (voice commands). It shows how multiple AI sub-domains integrate into one intelligent system.

Q8 **[12 Marks]**

(a) Discuss AI applications in five domains. (b) For each, state the technique. (c) Discuss one risk of deploying AI in a sensitive domain such as healthcare.

Sample Answer. **(a)/(b)** Healthcare — vision for scan diagnosis; finance — ML fraud detection; transport — vision + RL for driving; agriculture — ML yield/disease prediction; education — NLP tutoring. **(c)** In **healthcare**, a biased or poorly validated model could **misdiagnose** patients, and its "black-box" nature makes errors hard to explain or contest. Mitigation: rigorous testing on diverse data, human doctor oversight, and explainable models — treating the AI as a decision *aid*, not a replacement.

Q9 **[12 Marks]**

(a) Explain the philosophical and practical aspects of AI. (b) Explain two limitations of current AI. (c) Discuss the importance of responsible/ethical AI.

Sample Answer. **(a)** **Philosophical**: can machines think, understand or be conscious (Turing Test, Chinese Room, strong vs weak AI)? **Practical**: how to build reliable, useful systems. **(b)** Limitations: **data dependence and bias** (unfair data → unfair output) and **lack of understanding/common sense** (brittle on novel inputs). **(c)** **Responsible AI** matters because AI now affects jobs, justice, health and privacy; using fair data, ensuring transparency, protecting privacy and keeping humans in control prevent harm and build public trust — making ethics a core engineering concern, not optional.

Q10 **[12 Marks]**

(a) Explain the AI learning pathway. (b) Map each stage to prerequisite skills. (c) Explain how Units 1–5 of this course build toward AI.

Sample Answer. **(a)** Foundations → programming & maths → core AI (agents, search) → machine learning → deep learning/specialisation → projects. **(b)** Skills: logic and problem solving; Python, probability, linear algebra; data handling; ML theory; then domain depth. **(c)** **Unit 1** gives hardware and data representation; **Unit 2** problem solving and algorithms; **Unit 3** logic, tracing and testing; **Unit 4** AI concepts and agents; **Unit 5** machine learning. Each unit is a deliberate step up the pathway, turning a beginner into someone ready for advanced AI study.

Q11 **[12 Marks]**

(a) Explain intelligent behaviour and rationality. (b) Explain environment types with examples. (c) Explain how environment type affects agent design.

Sample Answer. **(a)** **Intelligent behaviour** is acting effectively toward goals; **rationality** is choosing actions that maximise expected performance given knowledge (bounded by time and data). **(b)** Environments may be fully/partially observable (chess vs poker), deterministic/stochastic (puzzle vs dice), static/dynamic (crossword vs driving), discrete/continuous (board vs road). **(c)** **Design impact**: partially observable worlds need internal state (memory); stochastic worlds need probability and planning under uncertainty; dynamic worlds need fast, real-time decisions. The harder the environment, the more advanced the agent — from simple reflex up to learning agents.

Q12 **[12 Marks]**

(a) Explain how AI relates to Machine Learning and Deep Learning with a diagram. (b) Give one example task for each layer. (c) Explain why ML has become the dominant approach to AI.

Sample Answer. **(a)** Nested circles: **AI** $\supset$ **ML** $\supset$ **Deep Learning**. **(b)** AI (a rule-based chess move); ML (spam detection learned from data); Deep Learning (recognising faces in photos with neural networks). **(c)** **ML dominates** because, for perception and pattern tasks, **learning from data** outperforms hand-coded rules, and today's **abundant data** and **GPU computing** make it practical and highly accurate. Hence most modern AI achievements are actually ML — which is why Unit 5 focuses on it.

Q13 **[12 Marks]**

(a) Discuss the transformative impact of AI on society. (b) Discuss four risks. (c) Recommend four principles for governing AI responsibly.

Sample Answer. **(a)** AI raises productivity and enables advances in health, agriculture, education, accessibility and science — a general-purpose "new electricity". **(b)** **Risks**: bias/discrimination, privacy erosion, job displacement, and misuse (deepfakes, autonomous weapons, misinformation). **(c)** **Principles**: (i) *fairness* — represent all groups; (ii) *transparency* — explainable, auditable systems; (iii) *privacy & consent* — protect personal data; (iv) *human oversight & accountability* — keep humans responsible for high-stakes decisions. These make AI's benefits broad and its harms controlled.

Q14 **[12 Marks]**

(a) Explain problem solving as search in AI. (b) Define state space, initial/goal state, and heuristic. (c) Explain the difference between uninformed and informed search with one example each.

Sample Answer. **(a)** Many AI problems are solved by **searching** a space of possibilities: the agent formulates the problem, searches for a sequence of actions leading to the goal, then executes it. **(b)** The **state space** is the set of all configurations; the **initial state** is where the agent starts; the **goal state(s)** satisfy the goal test; a **heuristic** is an informed estimate of the cost remaining to the goal. **(c)** **Uninformed** (blind) search, e.g. *breadth-first search*, explores without domain knowledge — complete but often slow. **Informed** (heuristic) search, e.g. A^*, uses a heuristic to head toward the goal — usually much faster. Example: finding a route on a map is slow with blind search but efficient with A* using straight-line distance as the heuristic.

Q15 **[12 Marks]**

(a) Summarise Unit 4. (b) Explain how AI concepts connect to Unit 5 (ML). (c) Explain the role of an AI/ML engineer using this unit's ideas.

Sample Answer. **(a)** We covered AI's **definition, four approaches, history, agents/PEAS, strong vs weak AI, applications, philosophy and ethics**. **(b)** Since most modern AI is **learning-based**, Unit 4's rational-agent and learning-agent ideas lead directly into **Unit 5**, where machines learn from data (supervised, unsupervised, reinforcement). **(c)** An **AI/ML engineer** defines the task (PEAS), chooses between traditional and learning approaches, builds and trains models, evaluates them, and deploys them **responsibly** — applying exactly the concepts of this unit to real problems.

5 Unit 5 — Introduction to Machine Learning

Syllabus. A. Role and significance of ML within AI; learning paradigms (supervised, unsupervised, reinforcement). **B.** Classification and clustering; basic supervised algorithms and their working principles. **C.** Applications of ML across domains. *(CO5–CO6, K2–K4)*

Section A — 2 Mark Questions

(Very short answers; attempt all fifteen)

Q1 **[2 Marks]**

Define Machine Learning.

Sample Answer. **Machine Learning (ML)** is a branch of AI in which computers learn patterns from data and improve at a task with experience, without being explicitly programmed with fixed rules.

Q2 **[2 Marks]**

Name the three main learning paradigms of ML.

Sample Answer. **Supervised learning**, **unsupervised learning** and **reinforcement learning**.

Q3 **[2 Marks]**

What is supervised learning?

Sample Answer. **Supervised learning** learns from **labelled** data — each example has a known correct output — to predict outputs for new inputs.

Q4 **[2 Marks]**

What is unsupervised learning?

Sample Answer. **Unsupervised learning** works on **unlabelled** data and discovers hidden structure or groups on its own.

Q5 **[2 Marks]**

What is reinforcement learning?

Sample Answer. **Reinforcement learning** has an agent learn by **trial and error**, taking actions and receiving rewards or penalties to maximise total reward.

Q6 **[2 Marks]**

Differentiate classification and regression.

Sample Answer. **Classification** predicts a **category** (e.g. spam/not spam); **regression** predicts a **number** (e.g. house price).

Q7 **[2 Marks]**

What is clustering?

Sample Answer. **Clustering** is an unsupervised task that groups similar data points together without using any labels.

Q8 **[2 Marks]**

Define a feature and a label in ML.

Sample Answer. A **feature** is an input attribute (e.g. height); a **label** is the known correct output for an example (e.g. "cat").

Q9 **[2 Marks]**

What does KNN stand for, and what is its core idea?

Sample Answer. **K-Nearest Neighbours**; it classifies a new point by a **majority vote** of its K closest labelled neighbours.

Q10 **[2 Marks]**

Name any two supervised learning algorithms.

Sample Answer. Any two of: **KNN**, **decision tree**, **linear regression**, **logistic regression**, **SVM**, **Naïve Bayes**, **random forest**.

Q11 **[2 Marks]**

What is overfitting?

Sample Answer. **Overfitting** is when a model memorises the training data (including noise) and so performs poorly on new, unseen data.

Q12 **[2 Marks]**

What is a training set and a test set?

Sample Answer. The **training set** is the data used to teach the model; the **test set** is separate unseen data used to check how well it generalises.

Q13 **[2 Marks]**

Name the popular clustering algorithm studied in this unit.

Sample Answer. **K-Means**, which groups data into K clusters around K centres.

Q14 **[2 Marks]**

State one application of ML in healthcare.

Sample Answer. **Disease detection** from medical images such as X-rays or MRI scans (any valid example).

Q15 **[2 Marks]**

Which paradigm is used to train a game-playing AI like AlphaGo?

Sample Answer. **Reinforcement learning** — the agent learns winning strategies through rewards from playing many games.

Section B — 4 Mark Questions

(Short answers with an example; attempt all fifteen)

Q1 **[4 Marks]**

Explain the role and significance of ML within AI.

Sample Answer. **ML** is the practical engine of modern AI: rather than hand-coding rules, it lets machines **learn from data**. It handles problems with no clear formula (vision, speech, language), **improves** with more data, **scales** to huge datasets and **adapts** to new situations. Because most recent AI successes are learning-based, ML sits at the heart of AI — which is why it forms this course's final unit.

Q2 **[4 Marks]**

Explain how a machine "learns" from data in four steps.

Sample Answer. (1) The model makes a **prediction**; (2) it compares the prediction with the correct answer and measures the **error**; (3) it **adjusts** its internal parameters to reduce the error; (4) it **repeats** over many examples until it performs well. Like a student practising and learning from mistakes, the model gradually improves — this loop is the essence of learning.

Q3 **[4 Marks]**

Compare supervised and unsupervised learning on four points.

Sample Answer. **Supervised**: labelled data, goal is to predict a known output, feedback via correct answers, e.g. spam detection. **Unsupervised**: unlabelled data, goal is to find hidden structure, no correct answers given, e.g. customer grouping. In short, supervised learning *predicts with a teacher*; unsupervised learning *discovers patterns without one.*

Q4 **[4 Marks]**

Explain the difference between classification and regression with one example each.

Sample Answer. Both are **supervised**. **Classification** outputs a *discrete category*: "is this email spam or not?". **Regression** outputs a *continuous number*: "what will this house cost?". Quick test: if the answer is a *label* it is classification; if a *number*, regression. Some algorithms (e.g. decision trees) can do both.

Q5 **[4 Marks]**

Explain clustering with a real example and name one clustering algorithm.

Sample Answer. **Clustering** groups similar unlabelled items so that points in a group are alike and points in different groups differ. **Example**: a shop groups customers by buying behaviour to target offers, without any predefined labels. A common algorithm is **K-Means**, which forms K clusters around K centres. It is an **unsupervised** task.

Q6 **[4 Marks]**

Explain the KNN algorithm's working in steps.

Sample Answer. **KNN**: (1) choose K; (2) compute the **distance** from the new point to all labelled points; (3) pick the K nearest; (4) assign the class by **majority vote** among them. It is a **lazy learner** — it stores the data and computes at prediction time. Choosing an odd K avoids ties.

Q7 **[4 Marks]**

Explain the train/test split and the golden rule of ML evaluation.

Sample Answer. Data is split into a **training set** (to learn) and a **test set** (to check on unseen data), often about 80%/20%. The **golden rule** is: **never test on the training data** — a model must be judged on data it has never seen, like a fair exam, otherwise its reported accuracy is misleading.

Q8 **[4 Marks]**

Explain overfitting and underfitting with a simple analogy.

Sample Answer. **Underfitting**: the model is too simple and misses the real pattern (poor on training *and* test). **Overfitting**: the model is too complex and memorises noise (great on training, poor on test). Analogy: a student who *understands* generalises to new questions (good fit); one who merely *memorises* answers (overfits) fails a new paper.

Q9 **[4 Marks]**

Describe how a decision tree makes a classification.

Sample Answer. A **decision tree** asks a series of yes/no questions on the most **informative** features, splitting the data at each node until it reaches a **leaf** that gives the class. Example: "weight $>$ 145? $\rightarrow$ colour $>$ 6?" leads to Apple/Orange. It is easy to read and explain, but a very deep tree can **overfit**.

Q10 **[4 Marks]**

List four real-world applications of ML across different domains.

Sample Answer. **Healthcare** — diagnosis from scans; **finance** — fraud detection; **e-commerce** — product recommendations; **transport** — self-driving assistance. (Also voice assistants, translation, spam filtering.) ML adds value wherever data is plentiful and rules are hard to hand-code.

Q11 **[4 Marks]**

Explain the significance of data quality in ML ("garbage in, garbage out").

Sample Answer. A model is only as good as its **data**. If the training data is wrong, incomplete or **biased**, the model learns those flaws and gives poor or unfair predictions — **garbage in, garbage out**. Hence collecting clean, representative and fair data is often more important than the choice of algorithm.

Q12 **[4 Marks]**

Differentiate supervised classification from clustering.

Sample Answer. **Classification** is *supervised*: it sorts data into **known** classes using labelled examples. **Clustering** is *unsupervised*: it **discovers** groups in unlabelled data. In short, classification places items into pre-defined boxes, while clustering *creates* the boxes itself from the data's structure.

Q13 **[4 Marks]**

Explain reinforcement learning using the agent–environment idea with an example.

Sample Answer. In **RL**, an **agent** takes an **action** in an **environment**, which returns a new state and a **reward** or penalty; the agent learns a **policy** that maximises total reward. **Example**: a game AI that gains points for good moves and loses them for bad ones gradually learns to win — as in training a dog with treats.

Q14 **[4 Marks]**

State four responsible-ML practices.

Sample Answer. Use **fair, representative** data; protect **privacy**; prefer **explainable** models for important decisions; keep **humans in control** and test carefully before deployment. These reduce bias and harm, ensuring ML is used ethically and safely.

Q15 **[4 Marks]**

Explain the relationship between AI, ML and Deep Learning.

Sample Answer. They are **nested**: **Deep Learning** ⊂ **Machine Learning** ⊂ **AI**. AI is the broad goal of intelligent behaviour; ML is the subset that learns from data; deep learning is the subset of ML using multi-layer **neural networks**. So all deep learning is ML, and all ML is AI, but not the reverse.

Section C — 6 Mark Questions

(Explanation with a diagram or a small worked example; attempt all fifteen)

Q1 **[6 Marks]**

Explain the three learning paradigms with a diagram and one example each.

Sample Answer. (Diagram: a central "ML" node branching to three boxes.) **Supervised** — learns from labelled data (spam vs not spam). **Unsupervised** — finds structure in unlabelled data (customer grouping). **Reinforcement** — learns by reward/penalty (game-playing AI). The paradigm is chosen by **what data you have** and **what you want**: answers given → supervised; only inputs → unsupervised; an environment to act in → reinforcement.

Q2 **[6 Marks]**

Explain the complete ML workflow with a diagram.

Sample Answer. (Diagram: Collect → Prepare → Train → Test → Deploy/Predict.) **Collect** relevant data; **prepare** it (clean and split into train/test); **train** a model on the training set; **test** it on unseen data to measure accuracy; **deploy** it and predict on real inputs, monitoring over time. The **train/test split** (e.g. 80/20) and the rule "never test on training data" ensure honest evaluation.

Q3 **[6 Marks]**

Work through a small KNN example: classify the point (155, 7) using $K = 3$ from the fruit data.

Sample Answer. Data: (150,8)=Apple, (170,7)=Apple, (130,6)=Orange, (140,5)=Orange. Compute distances to (155,7): to (150,8) ≈ 5.1; to (170,7)$= 15$; to (140,5)≈ 15.1; to (130,6)$= 25$. The three nearest are (150,8)=Apple, (170,7)=Apple, (140,5)=Orange. Majority vote: **Apple 2**, Orange 1 ⇒ predict **Apple**. This shows KNN's "you are like your neighbours" principle.

Q4 **[6 Marks]**

Explain the K-Means clustering algorithm step by step with a diagram.

Sample Answer. **K-Means steps**: (1) choose K; (2) place K **centres** randomly; (3) assign each point to its **nearest** centre; (4) move each centre to the **average** of its assigned points; (5) repeat steps 3–4 until the centres stop moving. (Diagram: two clusters of points, each with a × centre.) The result is K groups of similar points; the number K must be chosen in advance (e.g. via the elbow method).

Q5 **[6 Marks]**

Explain a decision tree with a diagram and state one advantage and one limitation.

Sample Answer. (Diagram: root question "Weight > 145?" branching Yes → Apple, No → "Colour > 6?" → Apple/Orange.) A **decision tree** asks yes/no questions on informative features until a leaf gives the class. **Advantage**: easy to read and explain (a "white box"). **Limitation**: a deep tree can **overfit** the training data; pruning or using a **random forest** helps.

Q6 **[6 Marks]**

Explain classification vs clustering in a table with examples and algorithms.

Sample Answer.

Aspect	Classification	Clustering
Learning	Supervised	Unsupervised
Data	Labelled	Unlabelled
Goal	Assign to known classes	Discover groups
Example	Spam vs not spam	Group customers
Algorithm	KNN, Decision Tree	K-Means

Classification sorts into **known** boxes; clustering **creates** the boxes from the data.

Q7 **[6 Marks]**

Explain overfitting, underfitting and good fit with a diagram and how to detect overfitting.

Sample Answer. (Diagram: data points with a smooth line = good fit and a wiggly line = overfit.) **Underfit**: too simple, poor on train and test. **Good fit**: captures the true pattern, works on new data. **Overfit**: too complex, memorises noise. **Detection**: compare training vs test accuracy — high training accuracy but low test accuracy signals overfitting. Remedies include more data, simpler models and regularisation.

Q8 **[6 Marks]**

Explain linear regression with a small example and a diagram.

Sample Answer. **Linear regression** fits the best straight line $y = mx + c$ through the data and predicts y for a new x. (Diagram: scattered points with a best-fit line.) Example: house price vs size — bigger houses cost more, so the line slopes up and predicts a price for a new size. The line is chosen to **minimise the total error** between predictions and actual values. It is a **regression** (number-predicting) algorithm.

Q9 **[6 Marks]**

Explain, with examples, how to match a real problem to the right paradigm and task.

Sample Answer. Ask: is the data labelled, and what is wanted? **Spam filter** — labelled emails, predict a class → *supervised classification.* **House price** — labelled prices, predict a number → *supervised regression.* **Customer groups** — no labels, find structure → *unsupervised*

clustering. **Game AI** — learn by acting → *reinforcement*. Correct matching is a key ML skill.

Q10 **[6 Marks]**

Explain five real-world ML applications, naming the paradigm and task for each.

Sample Answer. **Spam detection** — supervised classification; **house-price prediction** — supervised regression; **customer segmentation** — unsupervised clustering; **market-basket analysis** — unsupervised association; **self-driving control** — reinforcement learning. This shows how the same three paradigms cover a wide range of practical problems across domains.

Q11 **[6 Marks]**

Explain why choosing K matters in KNN, with the effect of small and large K.

Sample Answer. In **KNN**, K sets how many neighbours vote. A **small** K (e.g. 1) makes the model sensitive to **noise** — one odd neighbour can decide the class. A **large** K smooths predictions but may **blur** class boundaries and include distant, irrelevant points. A moderate, usually **odd**, K balances the two and avoids tie votes. K is typically tuned by testing several values.

Q12 **[6 Marks]**

Explain supervised algorithms at a glance (any five) with a one-line idea and task for each.

Sample Answer. **Linear regression** — fit a line to predict a number (regression). **Logistic regression** — predict a yes/no probability (classification). **KNN** — vote of nearest neighbours (classification). **Decision tree** — yes/no questions to a decision (both). **SVM** — find the best separating boundary (classification). (Also Naïve Bayes, random forest.) **No single algorithm is best** — the right choice depends on the data and problem.

Q13 **[6 Marks]**

Explain the limitations of ML with examples.

Sample Answer. **Data hungry** — needs large, good data; **GIGO** — bad data gives bad models; **bias** — unfair data yields unfair predictions (e.g. a biased hiring model); **black box** — deep models are hard to explain; **no understanding** — ML has no common sense and can fail on inputs unlike its training data. These limits mean ML outputs must be checked and used responsibly.

Q14 **[6 Marks]**

Explain how ML differs from traditional programming, with the "spam filter" example.

Sample Answer. **Traditional**: a programmer writes fixed rules ("if it contains 'lottery' mark spam"), which spammers easily evade. **ML**: we feed thousands of labelled emails and the model **learns** the patterns of spam itself, adapting as new spam appears. So the flow changes from *rules* → *output* to *data* → *rules (model)*. This is why ML wins for fuzzy, changing pattern problems.

Q15 **[6 Marks]**

Give a glimpse of deep learning: what it is, why it is powerful, and one example.

Sample Answer. **Deep learning** uses **neural networks** with many layers to learn complex patterns directly from raw data, learning its own features. (Diagram: input, hidden and output layers of connected nodes.) It is powerful for **vision, speech and language**, powering systems like image recognisers and ChatGPT, but it needs **lots of data** and computing power. It is studied in depth in later semesters; Unit 5 is the foundation.

Section D — 8 Mark Questions (Full explanation with diagram or complete worked example; attempt all fifteen)

Q1 **[8 Marks]**

Explain in detail the role and significance of ML within AI, and why ML "exploded" in the last decade.

Sample Answer. **ML** is the practical core of modern AI: instead of hand-coding rules, machines **learn from data**, which lets them tackle vision, speech and language where rules cannot be written by hand, and **improve, scale and adapt** automatically. Recalling the nested view (**Deep Learning** $\subset$ **ML** $\subset$ **AI**), most recent AI breakthroughs are actually ML. It **exploded after about 2012** because three things converged: **big data** (internet, phones, sensors), **computing power** (GPUs) and **better algorithms** (deep learning). Together they turned long-standing ideas into highly accurate, deployable systems.

Q2 **[8 Marks]**

Explain all three learning paradigms in depth with a comparison table and one detailed example each.

Sample Answer. **Supervised** learns input→output from labelled data; e.g. 10,000 emails labelled spam/not-spam train a filter. **Unsupervised** finds structure in unlabelled data; e.g. K-Means groups shoppers by behaviour. **Reinforcement** learns by reward; e.g. AlphaGo improves by playing and being rewarded for wins.

Aspect	Supervised	Unsupervised	Reinforcement
Data	Labelled	Unlabelled	Experience
Goal	Predict output	Find groups	Maximise reward
Feedback	Correct answers	None	Rewards/penalties

The paradigm follows the data and objective.

Q3 **[8 Marks]**

Explain classification and clustering in depth, with a diagram of each and a full comparison.

Sample Answer. **Classification** (supervised) learns a **decision boundary** from labelled data to place new points into known classes (diagram: two labelled point-clouds split by a line). **Clustering** (unsupervised) groups unlabelled points by similarity, discovering the groups (diagram: three point-clusters circled). **Comparison**: classification needs labels, targets known classes,

uses KNN/decision trees; clustering needs no labels, discovers groups, uses K-Means. **Key difference**: classification sorts into *known* boxes; clustering *creates* the boxes. Both reduce data to useful groupings but with very different starting information.

Q4 **[8 Marks]**

Explain the KNN algorithm fully and solve: classify (6, 4) with $K = 3$ given A(5,4)=Red, B(7,5)=Red, C(2,2)=Blue, D(8,1)=Blue.

Sample Answer. KNN: choose K, measure distances, take the K nearest, majority vote. Euclidean distances from (6,4): to A(5,4)= $\sqrt{1} = 1.00$; to B(7,5)= $\sqrt{2} \approx 1.41$; to D(8,1)= $\sqrt{4+9} = \sqrt{13} \approx 3.61$; to C(2,2)= $\sqrt{16+4} = \sqrt{20} \approx 4.47$. The three nearest are A (Red), B (Red), D (Blue). Majority vote: **Red 2**, Blue 1 $\Rightarrow$ predict **Red**. KNN is a lazy learner — no training, all work at prediction time; an odd K prevents ties.

Q5 **[8 Marks]**

Explain the K-Means algorithm in depth with a small numeric illustration of one iteration.

Sample Answer. K-Means: choose K; place centres; assign points to the nearest centre; move each centre to its points' mean; repeat until stable. **Illustration** ($K = 2$ on 1-D points $\{1, 2, 8, 9\}$, initial centres $c_1 = 1, c_2 = 9$): assign — $1, 2$ to c_1; $8, 9$ to c_2. Update — $c_1 = (1 + 2)/2 = 1.5$; $c_2 = (8 + 9)/2 = 8.5$. Re-assign gives the same groups, so the algorithm has **converged** to clusters $\{1, 2\}$ and $\{8, 9\}$. This shows how centres shift to the data's natural groups.

Q6 **[8 Marks]**

Explain overfitting and underfitting in depth: causes, detection and at least three remedies, with a diagram.

Sample Answer. (Diagram: an underfit straight line, a good-fit curve, and an overfit wiggly curve through noisy points.) **Underfitting** — model too simple; cause: too few features or too simple a model; poor on train and test. **Overfitting** — model too complex; cause: too many parameters or too little data; great on train, poor on test. **Detection**: a large gap between high training accuracy and low test accuracy. **Remedies**: (1) get **more/better data**; (2) use a **simpler model** or prune; (3) apply **regularisation**; (4) use **cross-validation**. The aim is a model that **generalises**.

Q7 **[8 Marks]**

Discuss ML applications across five domains in detail, naming the paradigm/task and benefit in each.

Sample Answer. Healthcare — supervised classification detects disease in scans, enabling early diagnosis. **Finance** — classification flags fraud in real time, cutting losses. **E-commerce** — recommendation (patterns/clustering) increases relevant sales. **Transport** — reinforcement + vision power driver assistance and routing. **Agriculture** — regression/classification predict yield and detect crop disease from images. In each, ML handles **scale and complexity** beyond fixed rules, but human oversight remains vital for trust and safety.

Q8 **[8 Marks]**

Explain the ML workflow end to end using a concrete project (predicting student pass/fail), covering data, features, model, evaluation and deployment.

Sample Answer. **Problem**: predict pass/fail (classification). **Collect** data: past students' attendance, marks, study hours with pass/fail labels. **Features**: attendance %, internal marks, hours; **label**: pass/fail. **Prepare**: clean and split 80/20. **Train** a classifier (e.g. decision tree) on the training set. **Evaluate** on the test set using accuracy (and check for bias). **Deploy**: predict for current students to flag those needing help, monitoring results over time. This shows the full pipeline from raw data to a useful, responsibly-used model.

Q9 **[8 Marks]**

Compare at least five supervised algorithms in a table (idea, task, one strength) and advise how to choose among them.

Sample Answer.

Algorithm	Idea	Task
Linear regression	Fit a line to predict a number	Regression
Logistic regression	Predict a yes/no probability	Classification
KNN	Vote of nearest neighbours	Classification
Decision tree	Yes/no question splits	Both
SVM	Best separating boundary	Classification
Random forest	Many trees vote	Both

Choosing: match the task (number vs class), the data size, the need for explainability (trees) vs accuracy (forests/SVM), and always **test** a few and compare — there is **no universally best** algorithm.

Q10 **[8 Marks]**

Explain reinforcement learning in depth with the agent–environment loop, key terms, and two applications.

Sample Answer. (Diagram: Agent $\xrightarrow{\text{action}}$ Environment $\xrightarrow{\text{reward, state}}$ Agent.) In **RL**, the **agent** observes a **state**, takes an **action**, and receives a **reward** and new state; it learns a **policy** (what to do in each state) that maximises **total reward** over time, balancing **exploration** (trying new actions) and **exploitation** (using known good ones). **Applications**: game-playing AI (AlphaGo) and robotics/self-driving control. RL suits sequential decision problems where feedback comes as rewards rather than labelled examples.

Q11 **[8 Marks]**

Discuss the limitations of ML and the principles of responsible ML in detail, with examples.

Sample Answer. **Limitations**: ML is **data-hungry**; suffers **GIGO**; can be **biased** (e.g. a hiring model trained on skewed data); is often a **black box**; and lacks **common sense**, failing on unfamiliar inputs. **Responsible-ML principles**: use **fair, representative data**; protect

privacy and obtain consent; prefer **explainable** models where decisions matter; keep **humans in control**; and **test** thoroughly for accuracy and fairness before and after deployment. Since a model reflects its data, responsibility begins with the data and continues through monitoring.

Q12 **[8 Marks]**

Explain feature representation and its importance, connecting it to abstraction from Unit 2, with an email-spam example.

Sample Answer. **Features** are the inputs a model learns from — numeric or categorical summaries of raw data. This is **abstraction** (Unit 2): keeping what matters, dropping the rest. For **spam**, raw email text becomes features like "number of links", "contains the word 'free' ", "sender reputation", "% capital letters". Good features make patterns **learnable** and strongly affect accuracy, while poor features hide the signal — often feature choice matters more than the algorithm. Thus abstraction learned in Unit 2 becomes a concrete, high-impact ML skill.

Q13 **[8 Marks]**

Explain the difference between classification and regression in depth, with two examples of each and how their evaluation differs.

Sample Answer. **Classification** predicts a **category**; examples: spam/not-spam, disease positive/negative. It is evaluated by **accuracy** (and precision/recall for imbalanced data). **Regression** predicts a **number**; examples: house price, tomorrow's temperature. It is evaluated by **error measures** such as mean squared error (how far predictions are from actual values). The core difference is the output type — discrete vs continuous — which also dictates the algorithm and the evaluation metric used.

Q14 **[8 Marks]**

Explain how ML fits into the broader AI picture and how Units 1–4 prepared for it, with examples.

Sample Answer. **ML** is the learning subset of **AI** (with **deep learning** inside ML). It builds on the whole course: **Unit 1** — data representation (all data is binary; ML works on numeric features); **Unit 2** — problem solving and **abstraction** (features are abstractions); **Unit 3** — logic, tracing and **testing** (used to debug ML code and evaluate models); **Unit 4** — intelligent and **learning agents** (an ML model is the learning element of an agent). Thus ML is not isolated — it integrates every earlier unit into systems that learn from data.

Q15 **[8 Marks]**

Explain accuracy and the confusion matrix for a classifier, and compute accuracy for a spam filter that gives TP= 40, TN= 45, FP= 5, FN= 10.

Sample Answer. A **confusion matrix** tabulates predictions vs truth: **TP** (spam correctly caught), **TN** (ham correctly passed), **FP** (ham wrongly marked spam) and **FN** (spam missed). **Accuracy** $= (TP + TN)/(TP + TN + FP + FN)$. Here $= (40 + 45)/(40 + 45 + 5 + 10) = 85/100 =$ **0.85** (85%). Accuracy alone can mislead on **imbalanced** data, so **precision** $= TP/(TP + FP) = 40/45 \approx 0.89$ and **recall** $= TP/(TP + FN) = 40/50 = 0.80$ give a fuller

picture — here the filter misses 20% of spam (recall), which may matter more than raw accuracy.

Section E — 12 Mark Questions

(Comprehensive multi-part answers; attempt all fifteen)

Q1 **[12 Marks]**

(a) Define ML and explain its role in AI. (b) Explain how a machine learns. (c) Explain why ML matters for an AI/ML engineer.

Sample Answer. **(a)** **ML** lets computers learn patterns from data and improve with experience; it is the practical engine of AI, handling tasks with no clear rules. **(b)** The model predicts, measures its **error**, **adjusts** parameters to reduce it, and repeats over many examples until it performs well. **(c)** An **AI/ML engineer** spends most of their work **building, training and evaluating** models; understanding how learning works lets them choose algorithms, diagnose overfitting, and deploy models responsibly — making ML the central professional skill.

Q2 **[12 Marks]**

(a) Explain the three paradigms with a diagram. (b) Give two examples of each. (c) Explain how to decide which paradigm a new problem needs.

Sample Answer. **(a)** Central ML branching to **supervised** (labelled), **unsupervised** (unlabelled) and **reinforcement** (reward-based). **(b)** Supervised: spam filter, house-price prediction; unsupervised: customer segmentation, topic grouping; reinforcement: game AI, robot control. **(c)** Ask two questions: *Is the data labelled?* If yes → supervised; if no → unsupervised. *Is there an environment to act in with rewards?* If yes → reinforcement. The data and the objective together determine the paradigm.

Q3 **[12 Marks]**

(a) Explain classification and clustering with diagrams. (b) Compare them on five points. (c) Give a real application of each and the algorithm used.

Sample Answer. **(a)** **Classification** — a boundary separates labelled classes; **clustering** — groups discovered in unlabelled data (diagrams as described). **(b)** Learning (supervised vs unsupervised), data (labelled vs unlabelled), goal (known classes vs discovered groups), classes known (yes vs no), algorithm (KNN/tree vs K-Means). **(c)** Classification: **spam detection** with a decision tree; clustering: **customer segmentation** with K-Means. The two solve different problems depending on whether labels exist.

Q4 **[12 Marks]**

(a) Explain the KNN algorithm. (b) Solve: classify (3, 3) with $K = 3$ given (1,1)=A, (2,2)=A, (4,4)=B, (5,5)=B, (3,1)=A. (c) Discuss the effect of changing K to 1 and 5.

Sample Answer. **(a)** KNN votes among the K nearest labelled points by distance. **(b)** Distances from (3,3): (2,2)= $\sqrt{2} \approx 1.41$(A); (4,4)= $\sqrt{2} \approx 1.41$(B); (3,1)= 2(A); (1,1)= $\sqrt{8} \approx 2.83$(A); (5,5)= $\sqrt{8} \approx 2.83$(B). Three nearest: (2,2)A, (4,4)B, (3,1)A ⇒ **A 2**, B 1 ⇒ predict **A**. **(c)** $K = 1$: nearest is a tie at 1.41 (A or B) — very sensitive to noise. $K = 5$: all points vote —

A appears 3 times ⇒ A, a smoother but less local decision. Choosing K trades noise-sensitivity for boundary sharpness.

Q5 **[12 Marks]**

(a) Explain the K-Means algorithm. (b) Run one full iteration on 1-D points $\{2, 4, 10, 12\}$ with $K = 2$ and initial centres 2 and 12. (c) State the final clusters and one limitation of K-Means.

Sample Answer. **(a)** K-Means assigns points to nearest centres and moves centres to cluster means, repeating until stable. **(b)** Initial $c_1 = 2, c_2 = 12$: assign $2, 4$ to c_1 and $10, 12$ to c_2; update $c_1 = (2 + 4)/2 = 3$, $c_2 = (10 + 12)/2 = 11$. Re-assign: same groups ⇒ converged. **(c)** Final clusters $\{2, 4\}$ **and** $\{10, 12\}$. **Limitation**: K must be chosen in advance and the result can depend on the **initial centres**, so different starts may give different clusters.

Q6 **[12 Marks]**

(a) Explain overfitting/underfitting with a diagram. (b) Explain train/test split and its purpose. (c) Give three techniques to reduce overfitting.

Sample Answer. **(a)** (Diagram: underfit line, good-fit curve, overfit wiggly curve.) Underfit = too simple; overfit = memorises noise; good fit = captures the real pattern. **(b)** The **train/test split** (e.g. 80/20) trains on one part and evaluates on unseen data; its purpose is an **honest** estimate of generalisation — never test on training data. **(c)** Reduce overfitting by: (1) more/better data; (2) a simpler model or pruning; (3) regularisation or cross-validation. The goal is a model that performs well on **new** data.

Q7 **[12 Marks]**

(a) Explain the working of a decision tree. (b) Build a small tree (described) to decide play/no-play from weather. (c) State one strength and one fix for its main weakness.

Sample Answer. **(a)** A **decision tree** splits data by the most informative feature via yes/no questions until a leaf gives the class. **(b)** Root "Outlook?": Sunny → "Humidity high?" (Yes → No-play, No → Play); Overcast → Play; Rain → "Windy?" (Yes → No-play, No → Play). **(c)** **Strength**: easy to read and explain. **Weakness**: overfitting on deep trees — fixed by **pruning** or using a **random forest** (many trees voting).

Q8 **[12 Marks]**

(a) Explain supervised algorithms (any five). (b) State the task each solves. (c) Explain the "no free lunch" idea in choosing algorithms.

Sample Answer. **(a)/(b)** Linear regression — predict a number (regression); logistic regression — yes/no probability (classification); KNN — neighbour vote (classification); decision tree — question splits (both); SVM — best boundary (classification); random forest — tree ensemble (both). **(c)** The **"no free lunch"** idea: **no single algorithm is best for all problems**; each has strengths and weaknesses depending on the data's size, shape and noise. So engineers **try several** and compare on a test set rather than assuming one is always superior.

Q9 **[12 Marks]**

(a) Explain reinforcement learning with the agent–environment loop. (b) Define state, action, reward, policy. (c) Give two applications and why RL suits them.

Sample Answer. **(a)** (Loop: Agent $\xrightarrow{\text{action}}$ Environment $\xrightarrow{\text{reward, state}}$ Agent.) The agent acts, gets feedback, and improves. **(b)** **State** — the current situation; **action** — a choice available; **reward** — the numeric feedback; **policy** — the strategy mapping states to actions. **(c)** **Game AI** (AlphaGo) and **robotics/self-driving**: both involve **sequential decisions** with delayed feedback and no labelled dataset, which is exactly what RL is designed for — learning good long-term strategies from rewards.

Q10 **[12 Marks]**

(a) Describe ML applications in five domains. (b) Name paradigm and task for each. (c) Discuss one ethical risk and its mitigation in one domain.

Sample Answer. **(a)/(b)** Healthcare — supervised classification (diagnosis); finance — classification (fraud); e-commerce — clustering/recommendation; transport — reinforcement (control); agriculture — regression (yield). **(c)** In **healthcare**, a **biased** dataset (e.g. few samples from some groups) can cause **misdiagnosis** for those groups; **mitigation**: collect diverse, representative data, validate across groups, use explainable models, and keep a doctor in the loop. Ethics must be built into the pipeline, not added later.

Q11 **[12 Marks]**

(a) Explain the full ML workflow. (b) Apply it to a house-price predictor. (c) State how you would evaluate the model and detect overfitting.

Sample Answer. **(a)** Collect → prepare → train → test → deploy. **(b)** **House-price predictor**: collect past sales with size, location, rooms, price; features = size/location/rooms, label = price; split 80/20; train a **linear regression**; predict prices for new houses. **(c)** **Evaluate** with an error measure (e.g. mean squared error) on the **test set**; **detect overfitting** by comparing training vs test error — a low training error but a high test error signals overfitting, addressed with more data or a simpler model.

Q12 **[12 Marks]**

(a) Explain how ML relates to AI and deep learning. (b) Give a task for each layer. (c) Explain why deep learning needs so much data and computing.

Sample Answer. **(a)** Nested: **Deep Learning** $\subset$ **ML** $\subset$ **AI**. **(b)** AI — a rule-based planner; ML — spam detection learned from data; deep learning — face recognition with neural networks. **(c)** **Deep learning** has **millions of parameters** and learns its own features from raw data, so it needs **large datasets** to fix those parameters without overfitting and **heavy computing** (GPUs) to process many examples through many layers. This is why it became practical only after big data and modern hardware arrived.

Q13 **[12 Marks]**

(a) Explain feature selection and data preparation. (b) Explain why they affect accuracy. (c) Give a worked mini-example of turning raw data into features.

Sample Answer. **(a)** **Data preparation** cleans, formats and splits data; **feature selection** chooses the most informative inputs and drops noise. **(b)** They affect accuracy because a model can only learn from the features it is given — irrelevant or messy features hide the signal, while good features expose it (often mattering more than the algorithm). **(c)** **Example**: raw record "2026-09-04, Rs. 1200, Delhi" → features: *month* = 9, *amount* = 1200, *city-code* = Delhi → 1. The raw text becomes clean numeric/categorical features a model can use.

Q14 **[12 Marks]**

(a) Compare traditional programming and ML with examples. (b) Explain when to use each. (c) Describe a hybrid system that combines both.

Sample Answer. **(a)** **Traditional**: human writes rules, exact output, low data (e.g. compute EMI). **ML**: learns rules from data, adapts, needs much data (e.g. detect fraud). **(b)** Use **traditional** when rules are clear and answers must be exact; use **ML** when rules are unknown/complex but data is plentiful. **(c)** A **hybrid** banking system uses *rules* for hard policies (block blacklisted accounts, enforce limits) and an *ML model* to **score** transactions for fraud risk — combining predictable control with learned pattern detection, as real systems do.

Q15 **[12 Marks]**

(a) Summarise Unit 5. (b) Explain how it completes the course from Unit 1 to Unit 5. (c) Describe the next steps a student should take toward becoming an AI/ML engineer.

Sample Answer. **(a)** Unit 5 covered ML's **role in AI**, the **three paradigms**, **classification vs clustering**, **supervised algorithms** (KNN, decision trees, regression, K-Means), **overfitting** and **applications** with responsible ML. **(b)** It completes the ladder: **Unit 1** hardware/data → **Unit 2** problem solving → **Unit 3** logic/testing → **Unit 4** AI/agents → **Unit 5** machines that learn. **(c)** **Next steps**: master **Python** and **maths** (probability, linear algebra), practise on real datasets, learn a library like scikit-learn, then study **deep learning** and build **projects**. The foundation from this course makes those next steps achievable.

6 Presentation Topics (Seminar / Group Activity)

Instructions. Each student (or group of 2–3) selects *one* topic and prepares a **6–8 minute presentation** (8–12 slides) with a short demo, diagram or real example. Topics are grouped by unit; pick across units for variety. Assessment: content accuracy, clarity of explanation, quality of the visual/example, and answers to questions. **Cite your sources.**

Unit 1 — Hardware Aspect of CSE

1. **The Journey of Computers** — from the abacus to modern supercomputers, with a visual timeline of the five generations.
2. **Inside the CPU** — how the control unit, ALU and registers cooperate to run a single instruction (fetch–decode–execute).
3. **Why Computers Use Binary** — number systems and how numbers, text, images and sound all become 0s and 1s.
4. **2's Complement and Computer Arithmetic** — how machines subtract using addition, with worked examples.
5. **From Machine Code to Python** — the evolution of programming languages and the role of assemblers, compilers and interpreters.

Unit 2 — Visual Thinking and Problem Understanding

6. **Computational Thinking in Everyday Life** — decomposition, pattern recognition, abstraction and algorithms applied to a real routine.
7. **A Picture is Worth a Thousand Words** — how the right visual tool makes a hard problem easy (with before/after examples).
8. **From Problem to Pseudocode** — the seven-step problem-solving approach demonstrated on a chosen problem.
9. **Mental Models vs External Representations** — why we sketch, list and tabulate while solving problems.
10. **Six Visual Tools, One Project** — planning a real project using mind maps, timelines, tables and storyboards together.

Unit 3 — Visualizing Processes and Logic

11. **Flowcharts that Tell a Story** — designing clear flowcharts for real processes, with the standard symbols.
12. **Truth Tables and Digital Logic** — how AND, OR, NOT and XOR build the decisions inside every computer.
13. **Decision Tables for Business Rules** — turning complicated "if" policies into complete, error-free tables.
14. **The Art of the Dry Run** — catching bugs by tracing code by hand, with a live example (e.g. reversing a number).
15. **Testing and Debugging** — designing normal, boundary and invalid test cases to find hidden logic errors.

Unit 4 — Introduction to Artificial Intelligence

16. **A Short History of AI** — key milestones, the two AI winters, and the current boom.
17. **The Turing Test and the Chinese Room** — can a machine truly think? A debate-style presentation.
18. **Intelligent Agents and PEAS** — specifying real agents (self-driving car, vacuum robot) using the PEAS framework.
19. **AI vs Traditional Programming** — when to hand-code rules and when to let the machine learn, with examples.
20. **Ethics of AI** — bias, privacy, jobs and accountability, and how to build AI responsibly.
21. **AI for India** — real applications in agriculture, healthcare, language and governance.

Unit 5 — Introduction to Machine Learning

22. **Three Ways Machines Learn** — supervised, unsupervised and reinforcement learning, with one demo each.
23. **KNN by Hand** — classifying a new point using nearest neighbours, worked live on slides.
24. **How K-Means Finds Groups** — clustering explained step by step with an animation or sketch.
25. **Decision Trees You Can Read** — building and explaining a small tree on a real dataset.
26. **Overfitting: When a Model Memorises** — the good-fit vs overfit problem and how to avoid it.
27. **Machine Learning All Around Us** — five ML applications across domains, each mapped to its paradigm and task.
28. **A Glimpse of Deep Learning** — what neural networks are and why they power modern AI (ChatGPT, image recognition).

Integrative / Advanced Topics (Optional, Cross-Unit)

29. **From Bits to Brains** — trace one example (e.g. recognising a digit) all the way from binary data (Unit 1) to a trained ML model (Unit 5).
30. **Build a Mini Project** — take a small real problem, visualise it (Unit 2), design its logic (Unit 3), and describe an AI/ML solution (Units 4–5).
31. **Responsible AI/ML** — combine the ethics of AI (Unit 4) with responsible-ML practices (Unit 5) into a checklist for engineers.
32. **Careers in AI & ML** — the skills, roadmap and roles in the AI industry, mapped to this course's five units.

End of Assignment Question Bank — CSAI1101, Fundamentals of Computing and Artificial Intelligence.

Post your doubts: https://www.gcjana.in/courses/shardauniversity/2601/fundamentals-of-computing-and-artificial-intelligence/#Post-doubts

Dr. Gopal Chandra Jana, Dept. of CSE, Sharda University.