Answer Key & Explanations: Introduction to Computer Languages

1. Multiple Choice Questions (MCQs) - Answers & Explanations

1. Answer: (b)
Explanation: A set of instructions used for communication.

2. Answer: (b)
Explanation: Computers require instructions in machine-readable format (they don't interpret natural languages directly).

3. Answer: (b)
Explanation: C++ is a high-level programming language used to write programs; binary and assembly are low-level.

4. Answer: (b)
Explanation: A compiler translates source code (human-readable) into machine-readable binary so the computer can execute it.

2. Short Answer Questions - Model Answers

5. Explain the similarity between human language rules and computer language rules.
Both human and computer languages rely on well-defined rules (grammar/syntax). In human languages these rules ensure sentences convey the intended meaning; in computer languages syntax and semantics ensure instructions are unambiguous so the computer can perform the exact operations. Without following these rules, communication fails (e.g., incorrect grammar or syntax errors).

6. Why are computer languages necessary for interacting with electronic devices?
Electronic devices require precise, unambiguous instructions. Computer languages allow humans to express tasks and algorithms in a structured way that can be translated into low-level instructions the device understands. They bridge the gap between human intent and machine execution, enabling automation and complex calculations.

7. What is source code, and why can’t computers execute it directly?
Source code is the set of human-readable instructions written in a programming language. Computers cannot execute source code directly because they only understand machine code (binary). Source code must be translated into machine-readable form (via compilers or interpreters) before execution.

8. Give an example of a simple task where a computer performs better than humans using a program.
Calculating large factorials (e.g., 120!) is tedious and error-prone for humans. A computer program can compute this quickly and accurately, handling large intermediate numbers and repetitive steps without fatigue.

3. Scenario-Based / Application Questions - Sample Answers

9. Factorial of 120: Why is a program more efficient?
Manual computation of 120! involves many repeated multiplications and large intermediate numbers. A program uses optimized algorithms and arbitrary-precision arithmetic to compute the result quickly and reliably. Additionally, programs avoid human errors and can reuse tested library functions (e.g., factorial functions), making them far more efficient.

10. Designing an ATM interface: How do programming languages help?

Programmers write the backend logic (account verification, transaction processing) and the front-end interface (language selection, input prompts) in programming languages. These programs handle security checks, communicate with banking servers, and present simple options to users. The interface abstracts complex operations so users can perform transactions without understanding underlying code.

11. Sequence from C++ code to execution:
1) Programmer writes source code in C++ (human-readable). 2) The source code is passed to a compiler which checks syntax and translates it into machine code (object files/executable). 3) The produced executable runs on the target machine; the CPU executes the machine instructions. 4) If needed, linkers and libraries are combined before execution. This pipeline ensures correct translation from human intent to machine actions.

12. ATM transaction fails: who handles the issue and role of programming?
When an ATM transaction fails, the end user reports to bank management/customer support. Backend teams and programmers investigate logs, identify bugs or network issues, and deploy fixes. Well-designed programs include error handling, logging, and retry mechanisms to maintain smooth interaction and aid troubleshooting.

4. Critical Thinking / Discussion - Suggested Points & Model Answers

13. Why is learning one programming language similar to learning a spoken language?
Mastering a programming language builds fluency: you learn syntax, idioms, and problem-solving patterns. Just like spoken language fluency improves communication, programming fluency improves the ability to express algorithms clearly and efficiently. Once fluent, learning additional languages becomes easier because core concepts (variables, control flow, data structures) transfer.

14. How do interfaces simplify complex operations? Examples:
Interfaces hide complexity by exposing only the necessary controls (buttons, forms). Example: ATM abstracts account protocols into simple steps (insert card, enter PIN, select amount). Another example: smartphone apps hide cloud sync, authentication, and background tasks behind intuitive touches and buttons.

15. Importance of translating code to machine-readable format:
Translation (compilation/interpretation) is essential because CPUs only understand binary. Without translation, human-readable source code would be meaningless to machines, preventing execution. If this step didn’t exist, we couldn’t run software, and all tasks would require manual low-level instruction, which is impractical.