Student name:

Computer Science Check List

Paper 1: Computer systems

1. Systems Architecture		
1.1.1 Architecture of the CPU	☑	☒
Purpose of the CPU:	☐	☐
Understand the **Fetch-Execute Cycle**:	☐	☐
Fetch: How the CPU retrieves instructions from memory.	☐	☐
Decode: Breaking down the instruction into parts the CPU can understand.	☐	☐
Execute: Carrying out the instruction, such as performing calculations or moving data.	☐	☐
Learn the role of CPU components:	☐	☐
Arithmetic Logic Unit (ALU): Manages mathematical and logical operations.	☐	☐
Control Unit (CU): Directs operations within the CPU.	☐	☐
Cache: A smaller, faster type of volatile memory for storing frequently used data.	☐	☐
Registers: Small storage locations within the CPU, including:	☐	☐
Memory Address Register (MAR): Holds the address of where data is fetched or stored.	☐	☐
Memory Data Register (MDR): Stores data fetched from memory.	☐	☐
Program Counter: Keeps track of the next instruction address.	☐	☐
Accumulator: Stores results of calculations.	☐	☐

1.1.2 CPU Performance	☑	☒
Factors Affecting Performance:	☐	☐
Clock Speed: Measure in GHz, impacts how many instructions a CPU can process per second.	☐	☐
Cache Size: Larger cache means more data can be stored close to the CPU, reducing wait times.	☐	☐
Number of Cores: More cores allow parallel processing of instructions, increasing efficiency.	☐	☐

1.1.3 Embedded Systems	☑	☒
Embedded Systems:	☐	☐
Understand that these are computers integrated into other devices (e.g., washing machines, cars).	☐	☐
Characteristics: Usually specialized, optimized for specific tasks, and have limited functions compared to general-purpose computers.	☐	☐

2. Memory and Storage		
1.2.1 Primary Storage (Memory)	☑	☒
RAM vs ROM:	☐	☐
RAM (Random Access Memory):	☐	☐
Volatile memory used for temporarily storing data that is actively being worked on.	☐	☐
Understand its role in running programs and how it affects multitasking.	☐	☐
ROM (Read-Only Memory):	☐	☐
Non-volatile memory that contains essential instructions for booting the computer.	☐	☐
Understand why ROM is essential for storing the firmware.	☐	☐
Virtual Memory:	☐	☐
Learn why virtual memory is needed when RAM is insufficient.	☐	☐
Understand the process of swapping data between RAM and a hard drive (HDD or SSD).	☐	☐

1.2.2 Secondary Storage	☑	☒
Types of Storage:	☐	☐
Optical (e.g., CDs, DVDs):	☐	☐
Good for long-term storage, cheap, but slower access times and lower capacity.	☐	☐
Magnetic (e.g., Hard Drives):	☐	☐
Used for bulk storage, higher capacity, but slower compared to solid-state drives.	☐	☐
Solid State (e.g., SSDs):	☐	☐
Fast, durable, but more expensive and typically with lower capacity than magnetic storage.	☐	☐
Storage Characteristics:	☐	☐
Capacity: Total amount of data the device can store.	☐	☐
Speed: How quickly data can be read from or written to the device.	☐	☐
Portability: Ease with which the device can be moved or transported.	☐	☐
Durability: How well the device withstands physical damage.	☐	☐
Reliability: Likelihood that the device will continue to function over time.	☐	☐
Cost: Expense associated with the device per unit of storage.	☐	☐

1.2.3 Data Units	☑	☒
Data Storage Units:	☐	☐
Understand units of data: **Bit, Nibble (4 bits), Byte (8 bits), Kilobyte (KB = 1,000 bytes), Megabyte (MB = 1,000 KB)**, and so on.	☐	☐
Practice converting between units and calculate the storage requirements for different files.	☐	☐

1.2.4 Data Storage and Binary	☑	☒
Binary and Hexadecimal Conversion:	☐	☐
Learn to convert positive denary numbers (0-255) to binary (up to 8 bits) and vice versa.	☐	☐
Understand binary addition and how overflow errors occur.	☐	☐
Convert binary to hexadecimal and vice versa.	☐	☐
Binary Shifts:	☐	☐
Understand how left and right binary shifts affect the value of binary numbers.	☐	☐

1.2.5 Data Storage Representation	☑	☒
Characters:	☐	☐
Understand how characters are represented in binary using character sets like **ASCII** and **Unicode**.	☐	☐
Images:	☐	☐
Learn how images are stored as a series of pixels in binary form.	☐	☐
Study the impact of **color depth** and **resolution** on image file size and quality.	☐	☐
Understand what **metadata** is and how it relates to image storage.	☐	☐
Sound:	☐	☐
Learn how sound is converted from analog to digital form through sampling.	☐	☐
Understand how **sample rate, duration, and bit depth** affect sound quality and file size.	☐	☐

1.2.6 Compression	☑	☒
Types of Compression:	☐	☐
Lossy Compression: Reduces file size by removing some data, useful for images, audio, and video but with some loss of quality.	☐	☐
Lossless Compression: Reduces file size without losing any data, used for text or other files where loss cannot be tolerated.	☐	☐
Understand when and why to use each type of compression.	☐	☐

3. Computer Networks, Connections, and Protocols

1.3.1 Networks and Topologies	☑	☒
Types of Networks:	☐	☐
LAN (Local Area Network): Small geographical area, typically within a single building.	☐	☐
WAN (Wide Area Network): Larger geographical area, connecting multiple LANs, such as the internet.	☐	☐
Network Performance:	☐	☐
Factors such as **number of connected devices** and **available bandwidth**.	☐	☐
Network Topologies:	☐	☐
Star Topology: All devices connected to a central hub; easy to manage but costly.	☐	☐
Mesh Topology: Every device is connected to every other; highly reliable but complex to install.	☐	☐

1.3.2 Wired and Wireless Networks	☑	☒
Wired (Ethernet) vs Wireless (Wi-Fi, Bluetooth):	☐	☐
Wired Networks: Generally faster, more secure, and stable but less portable.	☐	☐
Wireless Networks: More convenient, flexible, and easier to install but subject to interference.	☐	☐
Encryption:	☐	☐
Learn how data is encrypted to ensure secure transmission over networks.	☐	☐
IP and MAC Addressing:	☐	☐
IP Addressing: Unique identifier for devices on a network, understanding IPv4 vs IPv6.	☐	☐
MAC Addressing: Physical address assigned to network interfaces.	☐	☐
Protocols:	☐	☐
Study common protocols like **TCP/IP** (basis for internet communication), **HTTP/HTTPS** (web communication), **FTP** (file transfer), **SMTP** (email sending), etc.	☐	☐

4. Network Security

1.4.1 Threats to Computer Systems and Networks	☑	☒
Forms of Attack:	☐	☐
Malware: Viruses, worms, trojans—malicious software intended to damage or disrupt systems.	☐	☐
Social Engineering: Tactics like phishing that exploit human psychology to gain access to sensitive information.	☐	☐
Brute-Force Attacks: Trying all possible combinations to guess passwords.	☐	☐
Denial of Service (DoS) Attacks: Flooding a network with traffic to make services unavailable.	☐	☐
Data Interception and Theft: Unauthorized access to data as it is transmitted.	☐	☐
SQL Injection: Attacks that manipulate SQL queries to gain unauthorized access to databases.	☐	☐

1.4.2 Preventing Vulnerabilities	☑	☒
Prevention Methods:	☐	☐
Penetration Testing: Simulating attacks to identify vulnerabilities.	☐	☐
Anti-Malware Software: Protects against malicious software.	☐	☐
Firewalls: Control incoming and outgoing network traffic based on security rules.	☐	☐
User Access Levels: Restricting what users can do or access based on their role.	☐	☐
Passwords and Encryption: Protect data by making it unreadable without the correct key.	☐	☐
Physical Security: Protecting hardware from physical threats (e.g., locks, security cameras).	☐	☐

5. Systems Software		
1.5.1 Operating Systems	☑	☒
Purpose and Functionality:	☐	☐
User Interface: Allows users to interact with the computer (e.g., GUI, command line).	☐	☐
Memory Management: Allocates and tracks memory usage by various programs, enabling multitasking.	☐	☐
Peripheral Management: Manages communication between the CPU and external devices (e.g., keyboards, printers).	☐	☐
User Management: Controls user access, rights, and security settings.	☐	☐
File Management: Organizes and tracks files on the computer.	☐	☐

1.5.2 Utility Software	☑	☒
Functions:	☐	☐
Encryption Software: Protects data by converting it into an unreadable format.	☐	☐
Defragmentation: Reorganizes data on a disk to improve efficiency.	☐	☐
Data Compression: Reduces file sizes to save space and speed up transmission.	☐	☐

6. Ethical, Legal, Cultural, and Environmental Impacts		
1.6.1 Societal Impact	☑	☒
Digital Technology Impact:	☐	☐
Understand ethical issues (e.g., privacy concerns, digital divide).	☐	☐
Study legal issues (e.g., intellectual property, data protection laws).	☐	☐
Consider cultural impacts (e.g., changes in social interaction).	☐	☐
Explore environmental concerns (e.g., e-waste, energy consumption).	☐	☐
Relevant Legislation:	☐	☐
Data Protection Act 2018: Rules for handling personal data.	☐	☐
Computer Misuse Act 1990: Laws against unauthorized access to computers.	☐	☐
Copyright Designs and Patents Act 1988: Protects intellectual property.	☐	☐
Software Licenses: Differences between **open-source** and **proprietary software**.	☐	☐

Computer Science Check List

Paper 2: Computational thinking, algorithms and programming

7. Algorithms		
2.1.1 Computational Thinking	☑	☒
Key Principles:	☐	☐
Abstraction: Simplifying complex systems by focusing on essential details.	☐	☐
Decomposition: Breaking down problems into smaller, more manageable parts.	☐	☐
Algorithmic Thinking: Creating step-by-step solutions to problems.	☐	☐

2.1.2 Designing, Creating, and Refining Algorithms	☑	☒
Algorithm Design:	☐	☐
Identify inputs, processes, and outputs for problems.	☐	☐
Use **Pseudocode** and **Flowcharts** to represent algorithms.	☐	☐
Identify common errors and refine algorithms using **Trace Tables**.	☐	☐

2.1.3 Searching and Sorting Algorithms	☑	☒
Search Algorithms:	☐	☐
Binary Search: Efficiently finds an item in a sorted list.	☐	☐
Linear Search: Checks each item in a list until the target is found.	☐	☐
Sorting Algorithms:	☐	☐
Bubble Sort: Simple comparison-based sorting.	☐	☐
Merge Sort: Divides data into smaller lists, sorts them, and merges back.	☐	☐
Insertion Sort: Builds the final sorted list one item at a time.	☐	☐

8. Programming Fundamentals		
2.2.1 Programming Constructs	☑	☒
Basic Constructs:	☐	☐
Use of **Variables, Constants, Inputs, Outputs, Assignments** in programming.	☐	☐
Control flow using **Sequence, Selection (IF statements), Iteration (Loops)**.	☐	☐
Operators:	☐	☐
Understand and use **Comparison (==, !=, <, >, etc.)** and **Arithmetic (+, -, , /, MOD, DIV, etc.)* operators.	☐	☐

2.2.2 Data Types	☑	☒
Data Types:	☐	☐
Learn to use **Integer, Real, Boolean, Character, String**.	☐	☐
Understand **Casting**: Temporarily converting one data type to another when necessary.	☐	☐

2.2.3 Additional Programming Techniques	☑	☒
String Manipulation: Concatenation, slicing, and other operations.	☐	☐
File Handling:	☐	☐
Open, **Read**, **Write**, and **Close** files within programs.	☐	☐
Data Structures:	☐	☐
Use of **Arrays (1D and 2D)** to store multiple values.	☐	☐
Use of **Records** to store structured data.	☐	☐
SQL (Structured Query Language):	☐	☐
Learn to perform basic operations like **SELECT, FROM, WHERE** in SQL.	☐	☐
Sub-programs:	☐	☐
Understand and use **Functions** and **Procedures** to create modular code.	☐	☐
Learn about local and global variables, passing arrays in functions/procedures.	☐	☐
Random Number Generation:	☐	☐
Implement and use random numbers in programs.	☐	☐

9. Producing Robust Programs		
2.3.1 Defensive Design	☑	☒
Anticipating Misuse: Design programs to handle unexpected input or errors gracefully.	☐	☐
Input Validation: Ensure inputs are within expected parameters.	☐	☐
Maintainability:	☐	☐
Use of **Sub-programs** for modular code.	☐	☐
Naming Conventions: Clear and descriptive names for variables and functions.	☐	☐
Indentation: Properly format code for readability.	☐	☐
Commenting: Explain complex parts of the code for future reference.	☐	☐

2.3.2 Testing	☑	☒
Types of Testing:	☐	☐
Iterative Testing: Test during development.	☐	☐
Final/Terminal Testing: Test after development is complete.	☐	☐
Test Data:	☐	☐
Normal Data: Typical input that should be accepted.	☐	☐
Boundary Data: Edge cases that test limits.	☐	☐
Invalid Data: Incorrect input that should be rejected.	☐	☐
Error Identification:	☐	☐
Syntax Errors: Grammatical mistakes in code that prevent it from running.	☐	☐
Logic Errors: Code runs but produces incorrect results.	☐	☐

10. Boolean Logic

2.4.1 Boolean Logic	☑	☒
Logic Gates:	☐	☐
Learn the function and symbols for **AND, OR, NOT** gates.	☐	☐
Create and interpret **Truth Tables**.	☐	☐
Understand how to combine multiple gates in **Logic Diagrams**.	☐	☐

11. Programming Languages and IDEs

2.5.1 Programming Languages	☑	☒
High-level vs Low-level Languages:	☐	☐
High-level: Easier to write, read, and maintain (e.g., Python, Java).	☐	☐
Low-level: More control over hardware, but harder to use (e.g., Assembly language).	☐	☐
Translators:	☐	☐
Understand the need for **Compilers** (translate entire program before running) vs **Interpreters** (translate line-by-line).	☐	☐

2.5.2 Integrated Development Environment (IDE)	☑	☒
IDE Tools:	☐	☐
Learn the tools available in an IDE such as **Code Editors, Error Diagnostics, Run-time Environment, Translators**.	☐	☐
Practice using these tools to write, test, and debug programs efficiently.	☐	☐