2.1 Revision - Model Answers

2.1 Revision - Model Answers

1. Explain the concept of abstraction and describe how it is used in software development. Provide an example where abstraction simplifies a real-world system. (6 marks)

Answer:
Abstraction is the process of removing unnecessary details to focus on essential elements of a system or problem. (1) In software development, abstraction allows programmers to work at different levels, hiding low-level complexities such as hardware interactions and focusing on high-level design. (1) For example, in object-oriented programming, classes abstract real-world entities by representing only their key attributes and behaviors. (1)

A real-world example is a **map application**, which abstracts geographical data by showing key locations, roads, and landmarks while omitting irrelevant details such as small buildings and terrain texture. (1) This makes navigation simpler and more efficient. (1) Another example is an **ATM interface**, which abstracts complex banking operations by providing users with simple options like "Withdraw Money" without exposing the underlying database and transaction processes. (1)

2.1 Revision - Model Answers

2. A weather app displays temperature, humidity, and wind speed but does not show complex atmospheric calculations.

(a) Explain how this is an example of representational abstraction. (3 marks)
Answer:
Representational abstraction involves simplifying a complex system by displaying only the most relevant information. (1) The weather app abstracts the complex meteorological data and models by showing only key outputs like temperature, humidity, and wind speed. (1) It removes unnecessary raw data such as atmospheric pressure calculations, satellite imaging, and prediction models, making the information more user-friendly. (1)

(b) Why is it important for software systems to use representational abstraction? (3 marks)
Answer:
Representational abstraction **improves usability** by presenting only the most relevant data, reducing cognitive overload for users. (1) It **enhances efficiency**, as users can quickly interpret the simplified data without needing to process unnecessary information. (1) Additionally, it **improves system performance**, as less data needs to be processed and displayed, reducing computational load. (1)

2.1 Revision - Model Answers

3. A supermarket system categorizes all items (e.g., milk, apples, cereal) under "products."

(a) Describe how abstraction by generalisation is used in this scenario. (4 marks)
Answer:
Abstraction by generalisation involves grouping similar objects into broader categories. (1) In the supermarket system, individual items such as milk, apples, and cereal are classified under the general category of "products." (1) This allows for a structured and efficient inventory system, where each product type can share common attributes such as price, barcode, and stock levels. (1) By generalising products, the system can handle a large variety of items without needing unique code for each one. (1)

(b) Explain one benefit of using generalisation when designing a system. (2 marks)
Answer:
Generalisation **reduces redundancy**, as similar objects can be managed under a common category, making the system easier to maintain. (1) It also **improves scalability**, as new products can be added without requiring changes to the overall system structure. (1)

2.1 Revision - Model Answers

4. A car rental system allows customers to book vehicles online without knowing how the internal database processes the request.

(a) Explain how information hiding is applied in this system. (4 marks)
Answer:
Information hiding is the practice of concealing internal system details from users while providing only necessary functionality. (1) In the car rental system, customers interact with a simple web interface to book a car but do not see how the system processes availability, pricing, or database transactions. (1) The system handles database queries, payment processing, and customer validation behind the scenes. (1) This separation ensures that users do not need to understand or interact with the internal mechanics of the booking system. (1)

(b) Why is information hiding important in software development? (3 marks)
Answer:
Information hiding **enhances security**, as users cannot access sensitive data or system internals. (1) It **improves maintainability**, allowing developers to modify internal components without affecting the user interface. (1) It also **reduces complexity**, ensuring that users only interact with necessary functions, improving the overall experience. (1)

2.1 Revision - Model Answers

5. You are designing a ticket booking system. Before implementation, you define the inputs, processes, and outputs.

(a) Why is it important to identify inputs and outputs before writing the program? (4 marks)
Answer:
Identifying inputs and outputs ensures the system meets user needs by defining what data will be processed and what results will be produced. (1) It helps in designing a clear and structured algorithm, reducing errors and inefficiencies. (1) It ensures compatibility with other system components by standardising data formats. (1) Finally, it allows for thorough testing by defining expected outputs for given inputs, making debugging easier. (1)

(b) Give an example of an input, a process, and an output in this system. (3 marks)
Answer:
Input: User selects a movie and enters payment details. (1)
Process: The system checks seat availability and processes payment. (1)
Output: A digital ticket is generated and emailed to the user. (1)

2.1 Revision - Model Answers

6. A program calculates the square root of a number entered by the user.

(a) Explain why a precondition is necessary for this function. (3 marks)
Answer:
A precondition ensures that the function receives valid input before execution. (1) The square root function should only accept **non-negative numbers**, as negative inputs would result in errors in standard mathematical operations. (1) Using preconditions prevents crashes and ensures the program runs as expected. (1)

(b) Write a Python code snippet that includes a precondition to check for valid input. (3 marks)

```
import math

def calculate_square_root(n):
    if n < 0:
        return "Error: Input must be a non-negative number"
    return math.sqrt(n)

print(calculate_square_root(25))  # Output: 5.0
print(calculate_square_root(-4))  # Output: Error message
```

(1 mark for checking input, 1 mark for handling error, 1 mark for correct function implementation)

2.1 Revision - Model Answers

7. A password validation system checks if a password is at least 8 characters long and contains at least one number.

(a) Write a Python function that implements this logic. (4 marks)

```
def validate_password(password):
    if len(password) < 8:
        return "Invalid: Password too short"
    if not any(char.isdigit() for char in password):
        return "Invalid: Must contain at least one number"
    return "Valid password"

print(validate_password("password"))  # Invalid: Must contain a number
print(validate_password("pass1234"))  # Valid password
```

(1 mark for checking length, 1 mark for checking numbers, 1 mark for returning messages, 1 mark for correct syntax)

(b) Explain how logical operators are used in your solution. (3 marks)
Answer:
The **AND** operator ensures that both conditions (length check and number check) must be true for a valid password. (1) The function uses **NOT** to verify if a number is missing (`not any(char.isdigit())`). (1) These logical checks allow the program to return appropriate feedback to the user. (1)

2.1 Revision - Model Answers

8. A banking system allows multiple users to transfer money at the same time.

(a) Describe a potential race condition that could occur in this system. (4 marks)
Answer:
A race condition occurs when multiple transactions attempt to access and modify the same bank account balance simultaneously, leading to inconsistent results. (1) For example, if two users initiate withdrawals at the same time, both processes may check the balance before either completes, leading to an overdraft or incorrect final balance. (1) The system might read an outdated value before updating it, causing both transactions to withdraw more than the available funds. (1) This results in data corruption or unintended financial errors. (1)

(b) Explain how locking mechanisms or synchronization could prevent this issue. (4 marks)
Answer:
Locking mechanisms **prevent multiple processes from accessing the same data at the same time** by restricting access until a transaction is complete. (1) **Mutex (mutual exclusion)** ensures that only one transaction can modify an account balance at a time. (1) Synchronization techniques such as **database transactions (ACID properties)** guarantee that updates occur in a controlled manner, ensuring data integrity. (1) By using **atomic operations**, a system can ensure that the balance check and update happen as a single, uninterrupted action. (1)

2.1 Revision - Model Answers

9. A web server handles multiple user requests at the same time. Some tasks share resources, while others run independently.

(a) Explain the difference between concurrency and parallelisation. (4 marks)
Answer:
Concurrency allows multiple tasks to make progress at the same time by switching between them, but they do not necessarily run simultaneously. (1) It is used in single-core systems where tasks are managed through time-slicing. (1) **Parallelisation**, on the other hand, involves multiple tasks executing simultaneously on multiple processors or cores. (1) It improves performance for computationally heavy tasks by dividing them into smaller parts that run in parallel. (1)

(b) Give an example of when a web server might use each technique. (4 marks)
Answer:
A web server uses **concurrency** when handling multiple HTTP requests by switching between tasks, such as serving a webpage while waiting for a database query to return data. (1) This allows the system to remain responsive without requiring additional CPU cores. (1)
The server uses **parallelisation** when processing large amounts of data, such as encrypting multiple files or compressing images, by distributing tasks across multiple processors. (1)
This improves efficiency by utilizing available computational resources more effectively. (1)

2.1 Revision - Model Answers

10. A robotic vacuum cleaner must decide where to move next based on sensor data. The system must handle multiple tasks at once: detecting obstacles, planning movement, and adjusting suction power.

Identify and describe two computational thinking techniques used in this system. (6 marks)
Answer:
(1) Thinking Abstractly – The vacuum cleaner abstracts real-world obstacles by converting sensor data into a simplified map. (1) Instead of storing detailed representations of the environment, it processes just enough data to decide movement paths. (1) This reduces computational complexity and allows for efficient navigation. (1)

(2) Thinking Concurrently – The system manages multiple tasks at once, such as moving, adjusting suction, and detecting obstacles simultaneously. (1) By using concurrency, the vacuum cleaner can react to changing conditions without waiting for one task to complete before starting another. (1) This allows for smooth and efficient operation, optimizing cleaning performance. (1)

2.1 Revision - Model Answers

11. A self-driving car needs to navigate roads, detect objects, and plan routes while processing real-time data.

Identify and explain three computational thinking techniques that the car's software must use. (6 marks)
Answer:
(1) Thinking Abstractly – The car simplifies the complex real-world environment into digital representations, using **sensor data** to identify objects like pedestrians and traffic lights without processing unnecessary details. (1) This abstraction allows the system to focus on essential driving decisions. (1)

(2) Thinking Ahead – The car anticipates possible scenarios (e.g., predicting a pedestrian's movement or traffic light changes). (1) It evaluates multiple potential routes, ensuring safe and efficient driving. (1)

(3) Thinking Concurrently – The car must process multiple tasks at the same time, including **object detection, speed control, and navigation**. (1) By using concurrent processing, it can make split-second decisions without delays. (1)