Computer Science Paper 3

Practical

Model Paper 2025

Time Allowed: 2 hours 30 minutes
Total Marks: 120

You must answer on the question paper.

You must bring a soft pencil (preferably type B or HB), a clean eraser, and a dark blue or black pen. You may use a simple calculator if needed.

Before attempting the paper, write your name, candidate number, centre name, and centre number clearly in the designated spaces.

Instructions for Candidates

- Answer all questions.
- Write your answer to each question in the space provided.
- You must show all necessary working clearly.
- Do not use an erasable pen or correction fluid.
- Avoid writing over any barcodes printed on the paper.

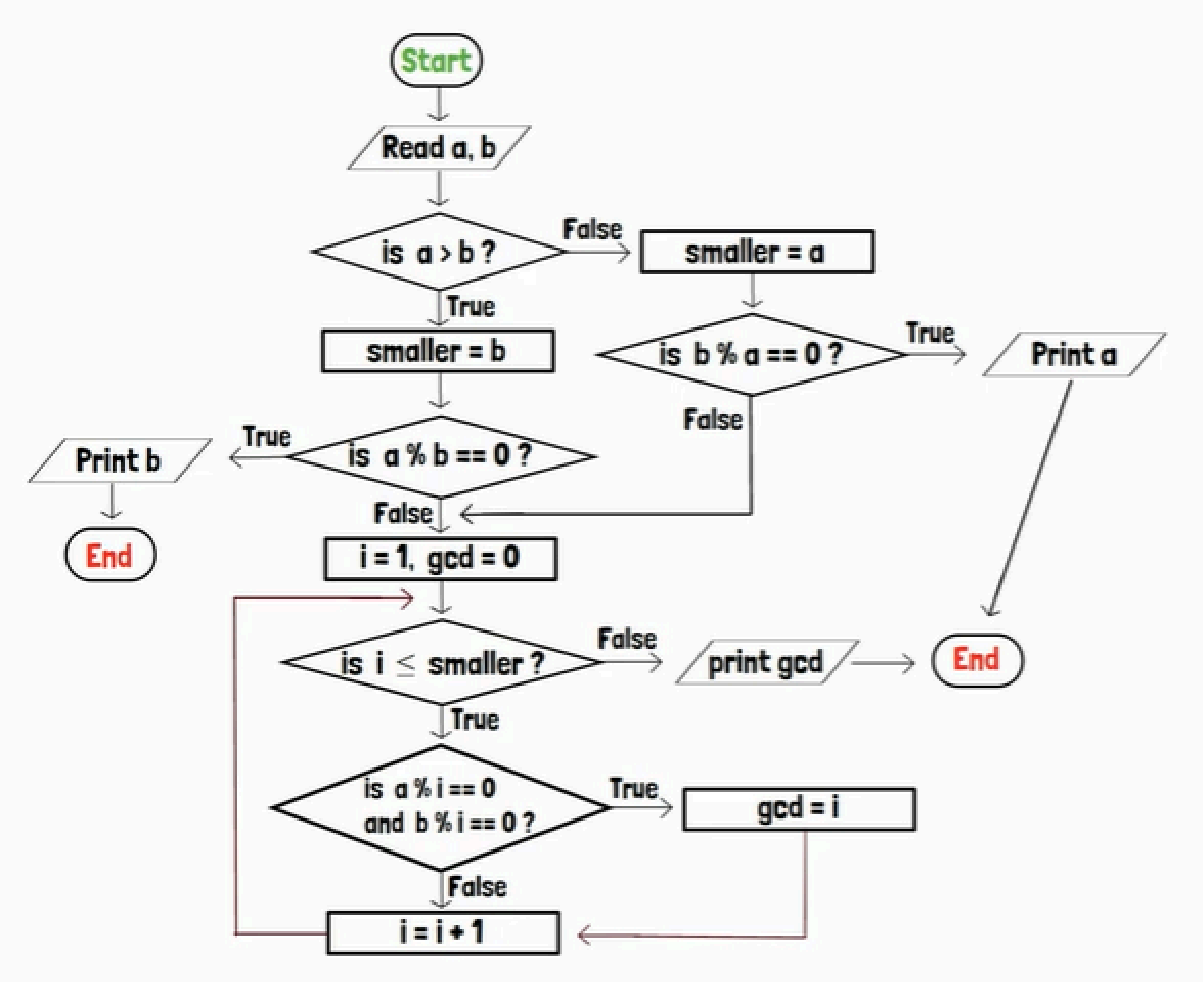
Information for Candidates

- This paper consists of a total of **120 marks**.
- The number of marks assigned for every question or its parts is indicated within brackets [ ].

Please read all questions carefully and follow the instructions exactly to ensure your responses are properly evaluated.

Q1.

The following flowchart represents a program segment that determines whether a number is prime.



(a) Write the corresponding pseudo code for this flowchart using if, for, and break statements. [6]

(b) You are required to design a binary search algorithm for an ascending list of numbers.

(i) Write the pseudo code of the algorithm. [3]

(ii) Trace the algorithm manually for the list:
[2, 4, 6, 8, 10, 12, 14, 16] to find the value 10.
Show all variable values (low, high, mid). [4]

(c) Two sorting algorithms are shown below.

Algorithm A (Bubble Sort)	Algorithm B (Insertion Sort)

for i in range(len(A)-1):
    for j in range(len(A)-1):
        if A[j] > A[j+1]:
            A[j], A[j+1] = A[j+1], A[j]
	for i in range(1, len(A)):
	    key = A[i]
	    j = i - 1
	    while j >= 0 and A[j] > key:
	        A[j+1] = A[j]
	        j -= 1
	    A[j+1] = key

i. State one difference between the working mechanism of Algorithm A and B. [2]

ii. Which algorithm is more efficient for nearly sorted data? Justify your answer. [2]

iii. Perform a single pass of Algorithm A for the list [9, 5, 2, 7]. Show the array after the pass. [3]

(d) The Fibonacci sequence can be defined recursively as:
F(0) = 0, F(1) = 1, F(n) = F(n-1) + F(n-2)

i. Write a recursive function in Python named Fibonacci (n) that prints the nth term. [4]

ii. Explain two advantages and two disadvantages of recursion compared to iteration. [4]

Q1 TOTAL MARKS: 28

Q2.

(a) A Python program is intended to calculate the average of 5 test scores but contains errors.

```
def average_score():
   total = 0
   for i in range(1,5):
      score = input("Enter score: ")
      total = total + score
   avg = total / 5
   print("Average score is: " + avg)
average_score()
```

i. Identify four errors in the code (syntax or logic). [4]

ii. Write the corrected program. [6]

(b) A teacher maintains a list of student names and their marks.

```
students = ["Ali", "Sara", "Bilal", "Hira"]
marks = [85, 92, 71, 66]
for i in range(5):
   print(students[i], ":", marks[i])
```

i. What runtime error will occur in this code? [2]

ii. Rewrite the **for loop** to correct the error and ensure all data displays correctly. [2]

iii. Modify the code to display only students who scored above 80. [3]

iv. Show the expected output. [3]

(c) You are given a file data.txt containing integers separated by spaces.
Write a Python code segment to:

1. Open the file and read its contents.
2. Display the largest and smallest numbers.
3. Close the file.

(Use appropriate file-handling syntax and comments.) [8]

(d) The following function is meant to check whether a given string is a palindrome (reads same forward and backward).

```
def palindrome(word):
    rev = ""
    for i in range(len(word)):
        rev = rev + word[i]
    if word == rev:
        return True
    else:
        return False
```

i. Identify the logical error. [2]

ii. Write the corrected function. [4]

iii. Predict the output of the corrected function for each input: [3]

- palindrome("level")
- palindrome("python")
- palindrome("madam")

iv. Suggest one test case for a boundary condition and justify your choice. [3]

Q2 TOTAL MARKS: 40

Q3.

A city's smart parking management system records available parking slots and car entries.
Each parking slot is represented by a class Slot with attributes:

SlotID, Status (Empty or Occupied), and VehicleNo.

(a) What do you understand by base class in OOP. [2]

(b) Define a Python class Slot with:

- A constructor to initialize all attributes.
- A method display_info() that prints the slot details neatly. [6]

(c) Write a function **assign_slot(slots, vehicle_no)** that:

- Finds the first Empty slot in the list slots.
- Assigns the vehicle number and changes the slot’s status to Occupied.
- Prints an appropriate message.

(Assume there are 10 slots in the parking system.) [8]

(d) Write another function **release_slot(slots, vehicle_no)** that:

- Searches for the vehicle number,
- Frees the slot (sets status to Empty), and
- Displays confirmation. [8]

(e) Represent the parking slots in a tabular format after assigning three cars: [8]

SlotID	VehicleNo	Status
S1	ABC-101	Occupied
S2	XYZ-235	Occupied
S3	LMN-412	Occupied
S4–S10	–	Empty

Write the Python list initialization to represent this structure.

Q3 TOTAL MARKS: 32

Q4.

(a) Assume the system records parking data over time and uses AI-based prediction to forecast busy hours. Explain two software testing methods suitable for verifying the Smart Parking System before deployment. [2]

(b) Explain with an example how a simple supervised machine learning approach (like linear regression or classification) could be integrated into this system to predict slot availability.
(Answer in 80–100 words, optionally with a labelled diagram or chart.) [10]

(c) Explain, with examples, how the Smart Parking System could use feedback from users and sensors after deployment to enhance performance and user experience. [8]

Q4 TOTAL MARKS: 20

Computer Science Paper 3

Answering key & Marking Scheme - Practical

Model Paper 2025

Time Allowed: 2 hours 30 minutes
Total Marks: 120

You must answer on the question paper.

You must bring a soft pencil (preferably type B or HB), a clean eraser, and a dark blue or black pen. You may use a simple calculator if needed.

Before attempting the paper, write your name, candidate number, centre name, and centre number clearly in the designated spaces.

Instructions for Candidates

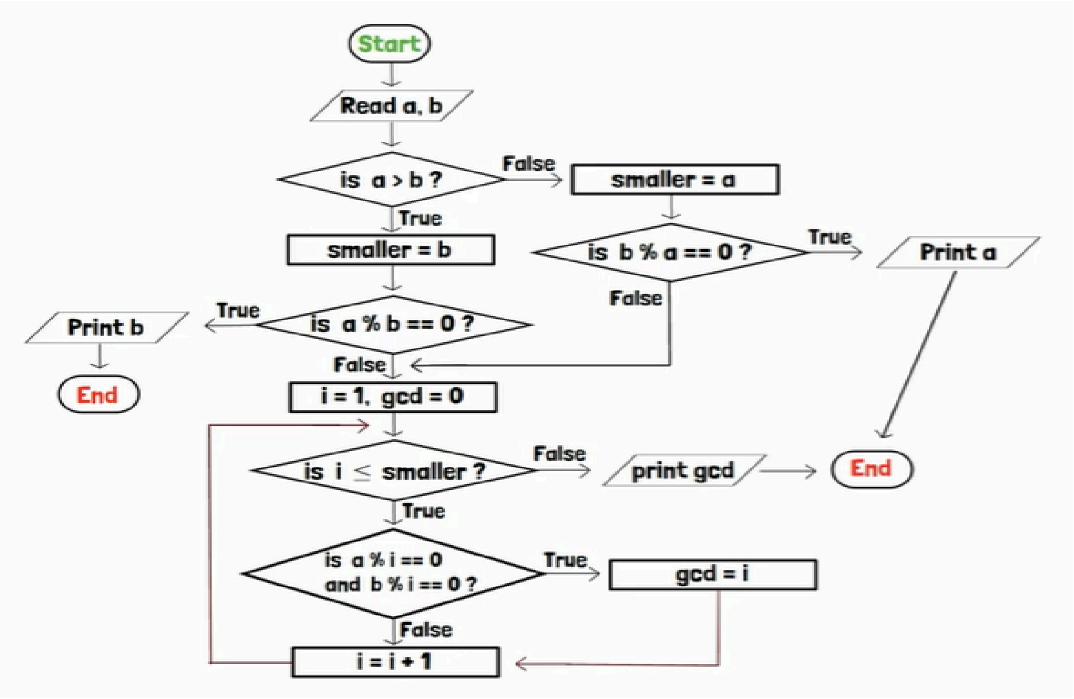
- Answer all questions.
- Write your answer to each question in the space provided.
- You must show all necessary working clearly.
- Do not use an erasable pen or correction fluid.
- Avoid writing over any barcodes printed on the paper.

Information for Candidates

- This paper consists of a total of **120 marks**.
- The number of marks assigned for every question or its parts is indicated within brackets [ ].

Please read all questions carefully and follow the instructions exactly to ensure your responses are properly evaluated.

Q1.The following flowchart represents a program segment that determines whether a number is prime.



(a) Write the corresponding pseudo code for this flowchart using if, for, and break statements. [6]

Answer:

```
BEGIN
  INPUT number
  flag ← 0
  FOR i ← 2 TO number/2 DO
     IF number MOD i = 0 THEN
        flag ← 1
     ENDIF
  ENDFOR
  IF flag = 0 THEN
     OUTPUT "Prime"
  ELSE
     OUTPUT "Not Prime"
  ENDIF
END
```

Marking Scheme:

- 1 mark for correctly accepting input.
- 2 marks for implementing the correct loop and divisor check.
- 2 marks for using the correct condition to identify a prime number.
- 1 mark for producing the correct output.

(b) You are required to design a binary search algorithm for an ascending list of numbers.

(i) Write the pseudo code of the algorithm. [3]

Answer:

```
BEGIN
  INPUT number
  flag ← 0
  FOR i ← 2 TO number / 2 DO
     IF number MOD i = 0 THEN
        flag ← 1
        BREAK
      ENDIF
    ENDFOR
    IF flag = 0 THEN
      OUTPUT "Prime"
    ELSE
      OUTPUT "Not Prime"
    ENDIF
  END
```

Marking Scheme:

Criterion	Description	Marks
Initialization	Correctly initializes low, high, and loop condition (WHILE low ≤ high).	1
Logic of mid and comparisons	Correct computation of mid and comparisons with search_value.	1
Update and output	Correct updating of low or high and proper output of result.	1

(ii) Trace the algorithm manually for the list:
[2, 4, 6, 8, 10, 12, 14, 16] to find the value 10.
Show all variable values (low, high, mid). **[4]**

Answer:

Step	low	high	mid	list[mid]	Comparison	Action
1	0	7	(0+7)/2 = 3	8	8 < 10 → search right half	low = 4
2	4	7	(4+7)/2 = 5	12	12 > 10 → search left half	high = 4
3	4	4	(4+4)/2 = 4	10	10 = 10	FOUND

Marking Scheme:

- 1 mark for setting the initial values of low, high, and mid.
- 1 mark for correctly updating mid in each iteration.
- 1 mark for performing the correct comparison and action.
- 1 mark for producing the correct final output.

(c) Two sorting algorithms are shown below.

Algorithm A (Bubble Sort)	Algorithm B (Insertion Sort)
for i in range(len(A)-1): **for j in range(len(A)-1):** **if A[j] > A[j+1]:** **A[j], A[j+1] = A[j+1], A[j]**	**for i in range(1, len(A)):** **key = A[i]** **j = i - 1** **while j >= 0 and A[j] > key:** **A[j+1] = A[j]** **j -= 1** **A[j+1] = key**

i. State one difference between the working mechanism of Algorithm A and B. **[2]**

Answer:

Aspect	Algorithm A – Bubble Sort	Algorithm B – Insertion Sort
Working mechanism	Repeatedly compares adjacent elements and swaps them until the largest element	Builds the sorted list one element at a time by inserting each new element into its

	"bubbles up" to its correct position after each pass.	correct position among already sorted elements.

Marking Scheme (2 marks):
- 1 mark for correctly describing the mechanism of Bubble Sort.
- 1 mark for correctly describing the mechanism of Insertion Sort.

ii. Which algorithm is more efficient for nearly sorted data? Justify your answer. [2]

Answer:

Algorithm B (Insertion Sort) is more efficient for nearly sorted data.

Justification:
Insertion Sort requires very few comparisons and shifts when elements are already close to their correct positions, resulting in a time complexity close to O(n)**.** In contrast, Bubble Sort still performs unnecessary comparisons and passes through the list, making it slower.

Marking Scheme:

- 1 mark for correctly identifying Insertion Sort.
- 1 mark for providing a logical justification (fewer comparisons/shifts for nearly sorted data).

iii. Perform a single pass of Algorithm A for the list [9, 5, 2, 7]. Show the array after the pass. [3]

Answer:

Algorithm A – Bubble Sort (Single Pass):
Compare adjacent elements and swap if the first is greater than the second.

Comparison	Elements Compared	Swap?	List after comparison
1	9 and 5	Yes	[5, 9, 2, 7]
2	9 and 2	Yes	[5, 2, 9, 7]
3	9 and 7	Yes	[5, 2, 7, 9]

Array after one pass**:** [5, 2, 7, 9]

Marking Scheme (3 marks):

- 1 mark for correctly showing pairwise comparisons.
- 1 mark for correctly indicating the swaps.
- 1 mark for showing the correct final list after the first pass.

(d) The Fibonacci sequence can be defined recursively as:
F(0) = 0, F(1) = 1, F(n) = F(n-1) + F(n-2)
i. Write a in Python named Fibonacci (n) that prints the nth term. [4]

Answer:

```
def Fibonacci(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return Fibonacci(n-1) + Fibonacci(n-2)
```

Marking Scheme:

- 1 mark for correctly handling the base case when n == 0.
- 1 mark for correctly handling the base case when n == 1.

- 1 mark for using the correct recursive call: Fibonacci(n-1) + Fibonacci(n-2).
- 1 mark for correct function structure and return/output statement.

ii. Explain two advantages and two disadvantages of recursion compared to iteration. **[4]**

Answer:

Advantages of Recursion:

1. Simpler logic and readability: Complex problems like tree traversal or Fibonacci sequence become easier to understand and implement recursively.
2. Reduces code length: Recursive functions often require fewer lines of code than equivalent iterative solutions.

Disadvantages of Recursion:

1. Higher memory usage: Each recursive call adds a new layer to the call stack, consuming more memory.
2. Slower execution: Recursive calls involve overhead due to repeated function calls and return operations, making them less efficient than iteration.

Marking Scheme:

- 1 mark each for two valid advantages.
- 1 mark each for two valid disadvantages.

Q1 TOTAL MARKS: 28

Q2.

(a) A Python program is intended to calculate the average of 5 test scores but contains errors.

```
def average_score():
    total = 0
    for i in range(1,5):
        score = input("Enter score: ")
        total = total + score
    avg = total / 5
    print("Average score is: " + avg)
average_score()
```

i. Identify four errors in the code (syntax or logic). **[4]**

Answer:

🎬 **The loop runs only 4 times**

`range(1,5)` collects 4 scores instead of 5.

🎬 **Input is not converted to a number**

`score = input()` returns a string but is added to an integer.

🎬 **String concatenation error**

`"Average score is: " + avg` → cannot add string and number.

🎬 **Logical error in average calculation**

Dividing by **5** even though only **4** scores are taken.

Marking Scheme:

- 1 mark for identifying that the loop runs only 4 times (range(1,5)).
- 1 mark for identifying that input() is not converted to a numeric type.
- 1 mark for identifying that the average calculation is wrong (divides by 5 while adding 4 scores).
- 1 mark for identifying that the print statement incorrectly concatenates a string with a number.

ii. Write the corrected program. **[6]**

Answer:

```
def average_score():
   total = 0
   num_scores = 5  # number of scores to collect
```

```
    for i in range(num_scores):
        score = float(input("Enter score: "))
        total += score
    avg = total / num_scores
    print("Average score is:", avg)

average_score()
```

Marking Scheme:

Criterion	Marks
1. Correct loop range to collect all 5 scores (e.g., `for i in range(5)` or using a variable like `num_scores = 5`)	1 mark
2. Correct conversion of input to numeric type (`int()` or `float()`)	1 mark
3. Correct accumulation of total using `total += score`	1 mark
4. Correct calculation of average using division by 5 (or variable `num_scores`)	1 mark
5. Correct print statement without string–number concatenation error (e.g., using comma or `str()`)	1 mark
6. Correct function structure and function call (`def average_score():` and `average_score()`)	1 mark

(b) A teacher maintains a list of student names and their marks.

```
students = ["Ali", "Sara", "Bilal", "Hira"]
marks = [85, 92, 71, 66]
for i in range(5):
    print(students[i], ":", marks[i])
```

i. What runtime error will occur in this code? [2]

Answer:

The program will give an IndexError: list index out of range because the loop goes to index 4, but both lists only have 4 items (last index is 3).

Marking Scheme:

- 1 mark for saying the error is IndexError.
- 1 mark for saying the loop goes past the list length.

ii. Rewrite the for loop to correct the error and ensure all data displays correctly. [2]

Answer:

```
students = ["Ali", "Sara", "Bilal", "Hira"]
marks = [85, 92, 71, 66]

for i in range(len(students)):
    print(students[i], ":", marks[i])
```

Marking Scheme:

- 1 mark for using `len(students)` to determine the loop range
- 1 mark for correctly displaying all student names with their marks

iii. Modify the code to display only students who scored above 80. [3]

Answer:

```
students = ["Ali", "Sara", "Bilal", "Hira"]
marks = [85, 92, 71, 66]
```

```
for i in range(len(students)):
    if marks[i] > 80:
        print(students[i], ":", marks[i])
```

Marking Scheme:
- 1 mark for using a loop to go through all students
- 1 mark for using an if condition to check `marks[i] > 80`
- 1 mark for correctly printing only the students with marks above 80

iv. Show the expected output. [3]

Answer:

```
Ali : 85
Sara : 92
```

Marking Scheme:
- 1 mark for including "Ali : 85"
- 1 mark for including "Sara : 92"
- 1 mark for showing only the students who scored above 80

(c) You are given a file data.txt containing integers separated by spaces.
Write a Python code segment to:

1. **Open the file and read its contents.**
2. **Display the largest and smallest numbers.**
3. **Close the file.**

(Use appropriate file-handling syntax and comments.) [8]

Answer:

```
# Open the file in read mode
file = open("data.txt", "r")

# Read the contents of the file
data = file.read()

# Split the contents into a list of numbers and convert to integers
numbers = [int(x) for x in data.split()]

# Find the largest and smallest numbers
largest = max(numbers)
smallest = min(numbers)

# Display the results
print("Largest number:", largest)
print("Smallest number:", smallest)

# Close the file
file.close()
```

Marking Scheme:
- 1 mark for opening the file correctly
- 1 mark for reading the file contents
- 2 marks for splitting the data and converting to integers
- 1 mark for finding the largest number
- 1 mark for finding the smallest number
- 1 mark for displaying the largest number correctly
- 1 mark for displaying the smallest number correctly
- 1 mark for closing the file

(d) The following function is meant to check whether a given string is a palindrome (reads same forward and backward).

```
def palindrome(word):
    rev = ""
    for i in range(len(word)):
        rev = rev + word[i]
    if word == rev:
        return True
    else:
        return False
```

i. Identify the logical error. **[2]**

Answer:

- The function does not reverse the string.
- `rev = rev + word[i]` adds characters in the original order instead of reverse order, so `rev` is the same as `word` and the palindrome check is incorrect.

Marking Scheme:

- 1 mark for identifying that the string is not reversed correctly
- 1 mark for explaining that the comparison with the original word will always be true for all strings

ii. Write the corrected function. **[4]**

Answer:

```
def palindrome(word):
    rev = ""
    for i in range(len(word)-1, -1, -1):  # Loop from end to start
        rev = rev + word[i]
    if word == rev:
        return True
    else:
        return False
```

Marking Scheme:

- 1 mark for initializing `rev` correctly
- 1 mark for using a loop that reverses the string (`range(len(word)-1, -1, -1)`)
- 1 mark for correctly accumulating characters in reverse order
- 1 mark for returning True if `word == rev` and False otherwise

iii. Predict the output of the corrected function for each input: **[3]**

- **palindrome("level")**
- **palindrome("python")**
- **palindrome("madam")**

Answer:

- `palindrome("level")` → True
- `palindrome("python")` → False
- `palindrome("madam")` → True

Marking Scheme:

- 1 mark for correctly predicting the output of `"level"`
- 1 mark for correctly predicting the output of `"python"`
- 1 mark for correctly predicting the output of `"madam"`

iv. Suggest one test case for a boundary condition and justify your choice. **[3]**

Answer:
• Test case: `palindrome("")` (empty string)
• Justification: An empty string is a boundary case because it has **zero length**. The function should handle it correctly and return `True`, as an empty string reads the same forward and backward.

Marking Scheme:
• 1 mark for suggesting a valid boundary test case
• 1 mark for explaining why it is a boundary case
• 1 mark for correct justification of the expected result

Q2 TOTAL MARKS: 40

Q3.

A city's smart parking management system records available parking slots and car entries.
Each parking slot is represented by a class Slot **with attributes:**

SlotID, Status (Empty or Occupied), and VehicleNo.

(a) What do you understand by base class in OOP. **[2]**

Answer:
• A base class (also called a parent class or superclass) is a class that provides common attributes and methods which can be inherited by other classes.
• It serves as a template for creating derived classes, allowing code **reuse** and establishing a hierarchical relationship in object-oriented programming.

Marking Scheme:
• 1 mark for defining a base class as a parent or superclass
• 1 mark for explaining that it provides attributes/methods that can be inherited

(b) Define a Python class **Slot with:**
- **A constructor to initialize all attributes.**
- **A method display_info() that prints the slot details neatly.** **[6]**

Answer:

```
class Slot:
 def __init__(self, SlotID, Status, VehicleNo):
        self.SlotID = SlotID
        self.Status = Status
        self.VehicleNo = VehicleNo
    def display_info(self):
        print("Slot ID:", self.SlotID)
        print("Status:", self.Status)
        print("Vehicle Number:", self.VehicleNo)
```

Marking Scheme:
• 1 mark for defining the class correctly
• 2 marks for writing the constructor (init) with all attributes
• 1 mark for initializing each attribute correctly
• 2 marks for defining display_info() that prints all slot details neatly

(c) Write a function **assign_slot(slots, vehicle_no) that:**
- **Finds the first Empty slot in the list slots.**
- **Assigns the vehicle number and changes the slot's status to Occupied.**

- **Prints an appropriate message.**

(Assume there are 10 slots in the parking system.) **[8]**

Answer:

```
def assign_slot(slots, vehicle_no):
    # Loop through all slots to find the first empty one
    for slot in slots:
        if slot.Status == "Empty":
            slot.VehicleNo = vehicle_no
            slot.Status = "Occupied"
            print(f"Vehicle {vehicle_no} has been assigned to Slot {slot.SlotID}.")
            return
    # If no empty slot is found
    print("No empty slots available.")
```

Marking Scheme:

- 2 marks for defining the function correctly with parameters `slots` and `vehicle_no`
- 2 marks for correctly finding the first Empty slot
- 2 marks for assigning the vehicle number and updating the status
- 1 mark for printing an appropriate success message
- 1 mark for handling the case when no empty slot is available

(d) Write another function release_slot(slots, vehicle_no) that:

- **Searches for the vehicle number,**
- **Frees the slot (sets status to Empty), and**
- **Displays confirmation.** **[8]**

Answer:

```
def release_slot(slots, vehicle_no):
    # Loop through all slots to find the vehicle
    for slot in slots:
        if slot.VehicleNo == vehicle_no:
            slot.VehicleNo = None
            slot.Status = "Empty"
            print(f"Vehicle {vehicle_no} has been released from Slot {slot.SlotID}.")
            return
    # If vehicle number is not found
    print(f"Vehicle {vehicle_no} not found in any slot.")
```

Marking Scheme:

- 2 marks for defining the function correctly with parameters `slots` and `vehicle_no`
- 2 marks for correctly searching for the vehicle number
- 2 marks for freeing the slot (setting status to Empty and clearing vehicle number)
- 1 mark for printing a confirmation message
- 1 mark for handling the case when the vehicle number is not found

(e) Represent the parking slots in a tabular format after assigning three cars: **[8]**

SlotID	VehicleNo	Status
S1	ABC-101	Occupied
S2	XYZ-235	Occupied
S3	LMN-412	Occupied
S4–S10	–	Empty

Write the Python list initialization to represent this structure.

Answer:

```
slots = [
   Slot("S1", "Occupied", "ABC-101"),
   Slot("S2", "Occupied", "XYZ-235"),
   Slot("S3", "Occupied", "LMN-412"),
   Slot("S4", "Empty", None),
   Slot("S5", "Empty", None),
   Slot("S6", "Empty", None),
   Slot("S7", "Empty", None),
   Slot("S8", "Empty", None),
   Slot("S9", "Empty", None),
   Slot("S10", "Empty", None)
]
```

Marking Scheme:

- 2 marks for correctly representing the first three occupied slots with vehicle numbers
- 2 marks for correctly representing the remaining empty slots
- 2 marks for correct use of Slot class constructor for initialization
- 2 marks for matching SlotID, VehicleNo, and Status accurately

Q3 TOTAL MARKS: 32

Q4.

(a) Assume the system records parking data over time and uses AI-based prediction to forecast busy hours.

Explain two software testing methods suitable for verifying the Smart Parking System before deployment. [2]

Answer:

- **Unit Testing:** This method tests individual components or functions of the system (e.g., assigning/releasing slots, data recording, prediction algorithms) to ensure each part works correctly.

- **System Testing:** This method tests the complete integrated system, including AI-based predictions, slot management, and user interface, to verify that the Smart Parking System functions as expected in real-world scenarios.

Marking Scheme:

- 1 mark for correctly explaining unit testing
- 1 mark for correctly explaining system testing

(b) Explain with an example how a simple supervised machine learning approach (like linear regression or classification) could be integrated into this system to predict slot availability. (Answer in 80–100 words, optionally with a labelled diagram or chart.) [10]

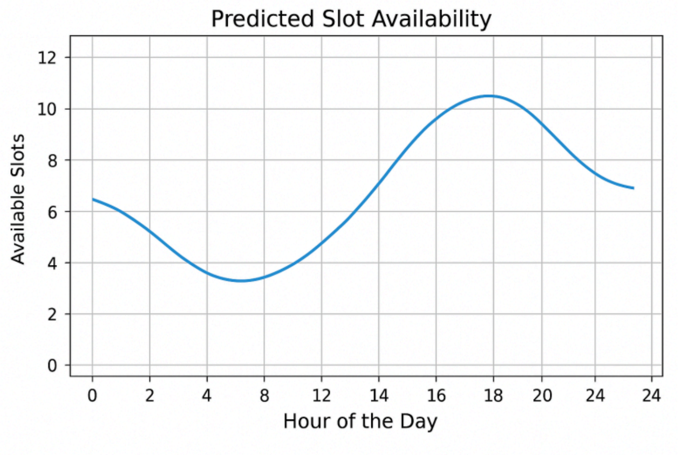
Answer:

A simple supervised learning approach can be used to predict parking slot availability based on historical data. For example, a linear regression model can be trained using past data with features like time of day, day of week, and number of cars entering/exiting as inputs, and available slots as the target. The model learns patterns to forecast busy hours and expected free slots.

Example:

Input: `hour=9, day=Monday, current_occupied=70%`

Output: Predicted available slots = 15



Marking Scheme:

- 3 marks for explaining supervised learning in context
- 3 marks for identifying features and target variable
- 2 marks for providing a clear example of prediction
- 2 marks for optional diagram/chart or clear description

(c) Explain, with examples, how the Smart Parking System could use feedback from users and sensors after deployment to enhance performance and user experience. [8]

Answer:

After deployment, the Smart Parking System can collect feedback from users (e.g., satisfaction surveys, complaints about finding slots) and sensor data (e.g., occupancy sensors, entry/exit counts). This data can be used to improve slot allocation algorithms**,** optimize parking predictions, and adjust dynamic pricing.

Examples:

- If users report long waits at certain hours, the system can predict busy periods and suggest alternative slots.
- Sensor data showing frequent empty slots in some areas can be used to reassign vehicles efficiently**.**
- Collecting feedback on app usability can help enhance the user interface for smoother navigation and notifications.

Marking Scheme:

- 2 marks for explaining the role of user feedback
- 2 marks for explaining the role of sensor data
- 2 marks for giving examples related to prediction and slot allocation
- 2 marks for giving examples related to improving user experience or interface

Q4 TOTAL MARKS: 20