Trial Questions:

(1)

You are required to develop an interactive **Rock, Paper, Scissors** game in Python with the following requirements:

1. The game should randomly choose between “rock”, “paper”, and “scissors” as the computer's choice for each round.
2. The user should be prompted to enter their choice ("rock", "paper", or "scissors").
3. Validate the user's input to ensure it's one of the three valid choices. If the input is invalid, prompt the user again until a valid option is provided.
4. Determine the winner based on the following rules:
   * Rock beats Scissors
   * Scissors beats Paper
   * Paper beats Rock
5. Display the result of the round, e.g., “You win!”, “Computer wins!”, or “It’s a tie!”
6. After each round, ask the user if they want to play again (yes or no). Only these responses should be accepted.
7. If the user enters anything other than “yes” or “no”, prompt again until a valid answer is given.
8. The game should continue to loop until the user chooses to quit.
9. Include error handling to manage unexpected inputs, including ensuring the program doesn’t crash if a non-string or unusual input is entered.

(2)

You are required to develop an interactive **Word Guessing Game** in Python with the following specifications:

1. The game should randomly select a secret word from a predefined list of words (e.g., ["apple", "banana", "grape", "orange", "melon"]).
2. The user should be allowed a maximum of **five attempts** to guess the correct word.
3. If the guessed word is alphabetically higher than the secret word, the program should display: "Your guess comes after the word!"
4. If the user correctly guesses the secret word, the program should display: "Congratulations! You guessed the word in X attempt(s)!" where X is the attempt number.
5. If the user fails to guess the word in all five attempts, the program should display: "Game over! The correct word was 'orange'." (use the actual correct word, not always 'orange').
6. The game must prompt the user at the end of each round with: "Do you want to play again? (yes or no):" and accept only 'yes' or 'no' as valid inputs.
7. If the user enters anything other than 'yes' or 'no', the program should prompt again until a valid input is provided.
8. The game should continue to loop until the user enters 'no' to quit.
9. Include error handling to ensure the user cannot crash the program by entering invalid input types (e.g., numbers instead of strings).

(3)

Write a Python program that simulates a basic password‐check with these requirements:

1. **Fixed Password**
   * Store a single secret password as a plain string in your code (e.g. "letmein").
2. **Guess Attempts**
   * Allow the user up to **3 attempts** to enter the correct password.
3. **Feedback**
   * After each wrong guess, print

Incorrect password. You have X attempt(s) left.

* + If they guess correctly, print

Access granted!

and skip the remaining attempts.

1. **Failure Message**
   * If all 3 attempts are used without success, print

Access denied. The correct password was: letmein

1. **Replay Prompt**
   * After the round ends (whether success or failure), ask the user:

Play again? (yes or no):

* + Only accept "yes" or "no" (case‐insensitive). If they type anything else, prompt again until valid.

1. **Looping**
   * Repeat the guess‐round until the user answers “no” to the replay prompt.
2. **Input Validation**
   * If, for some reason, the user presses Enter without typing anything, treat it as a wrong guess (don’t crash).

MATLAB

(4)

Develop an interactive MATLAB program that lets users reserve hotel rooms with various options, pricing tiers, and discounts. Your program should handle dynamic user input, validate entries, apply surcharges and discounts, and loop until the user decides to quit.

**Task Requirements:**

**(A) Display Available Room Types & Base Rates**

1. Present a predefined list of room types with their nightly base rates, for example:
   * Standard Room – $100/night
   * Deluxe Room – $150/night
   * Suite – $250/night
2. Allow users to view this menu before making a reservation.

**(B) Room Selection & Duration**

1. Prompt the user to select a room type by entering its corresponding number.
2. Ask for the number of nights they wish to stay.
3. Validate both inputs to ensure the selection exists and the nights are a positive integer.

**(C) Add-On Services**

1. Offer optional add-ons:
   * Breakfast Buffet – $15 per person per night
   * Airport Shuttle – flat $40 per stay
   * Extra Bed – $30 per night
2. Allow the user to choose any combination (or none) of these services.
3. Implement error checks for invalid selections.

**(D) Membership Discounts**

1. Prompt the user to enter membership status:
   * **None**: 0% discount
   * **Silver**: 5% discount
   * **Gold**: 10% discount
2. Use if-elseif-else or switch to apply the proper discount rate to the subtotal.

**(E) Payment Method Selection**

1. Provide choices for payment:
   * Cash
   * Credit Card (2% processing fee)
   * Mobile Wallet
2. Validate that the user selects one of the allowed methods and compute any payment fees.

**(F) View Option Surcharge**

1. Allow selection of room view:
   * City View – no extra charge
   * Sea View – additional $20 per night
2. Validate input and add the surcharge if Sea View is selected.

**(G) Booking Summary & Cost Calculation**

1. Calculate the total cost by summing:
   * Base room rate × nights
   * Add-on service charges
   * View surcharge
   * Payment processing fee
2. Apply membership discount to the subtotal before payment fees, then add fees.
3. Display a formatted summary:
   * Room type & view
   * Number of nights
   * Selected add-ons
   * Membership level & discount rate
   * Payment method & any fees
   * Total cost before discount, discount amount, and final payable amount

**(H) Looping Until Exit**

1. After showing the booking summary, ask the user:

Would you like to make another booking? (yes or no):

1. Only accept “yes” or “no” (case-insensitive). Re-prompt on invalid input.
2. Continue looping to allow multiple bookings until the user selects “no.”
3. On exit, display a friendly thank-you message and terminate the script.