Python Programming – Monthly Examination

Total Duration: 1 hour 20 minutes

Total Marks: 100

Allowed Topics: Variables, Operators, Conditional Statements, For Loops, Lists, Tuples

Question 1: Student Result Management System (Marks: 35)

Problem Statement:

Write a Python program to manage and analyze student marks.

Requirements:

• Input the total number of students.

• For each student, take their name and marks (out of 100).

• Store names and marks in lists.

• After taking all inputs: display all students with their marks, find and display highest marks, lowest marks, and average marks.

• Assign grade using if-else conditions as follows: Marks ≥ 80 → Grade A, Marks 60–79 → Grade B, Marks 40–59 → Grade C, Marks < 40 → Grade F.

Sample Output:

Enter number of students: 3 Enter name of student 1: Ali Enter marks of Ali: 85 Enter name of student 2: Sara Enter marks of Sara: 72 Enter name of student 3: Hamza Enter marks of Hamza: 39 --- Student Results --- Ali - 85 - Grade A Sara - 72 - Grade B Hamza - 39 - Grade F Highest Marks: 85 Lowest Marks: 39 Average Marks: 65.3

Question 2: Grocery Billing System (Marks: 35)

Problem Statement:

Design a Python program to calculate a simple grocery bill for a customer.

Requirements:

• Store 5 grocery items and their prices using tuples.

• Display all available items with their prices.

• Ask the user how many items they want to buy.

• For each selected item: input the item name and quantity, use a for loop to calculate subtotal (price × quantity).

• After all items are entered: display a bill summary showing each item, its quantity, and subtotal; calculate and display the total bill; apply a 10% discount if total bill > 1000; display the final payable amount.

Sample Output:

--- Available Items --- Apple - 200 Banana - 60 Rice - 150 Milk - 120 Bread - 100 How many different items do you want to buy? 2 Enter item name: Rice Enter quantity: 3 Enter item name: Milk Enter quantity: 2 --- Bill Summary --- Rice (3) = 450 Milk (2) = 240 Total Bill = 690 No Discount Applied Final Payable = 690

Question 3: Number Guessing Game (Marks: 30)

Problem Statement:

Create a simple number guessing game using basic programming logic.

Requirements:

• Store a secret number (for example, secret = 7) in a variable.

• Give the user 5 chances to guess the number.

• For each guess: if correct, print 'Correct Guess!' and stop the loop; if less than secret, print 'Too Low!'; if greater, print 'Too High!'.

• If all chances are used without guessing correctly, display 'Game Over! The number was 7.' Sample Output:

Guess the number (between 1–10): 3 Too Low! Guess the number (between 1–10): 8 Too High! Guess the number (between 1–10): 7 Correct Guess!

End of Paper