SALIM HABIB UNIVERSITY

Course: Programming Fundamentals	**Course Code:** CSC- 105
Lab Instructor: Mansoor Ahmed	**Date:** 28/11/2024 ‖ **Points:** 04

Lab: 06 - Lab Tasks

Scenario 1: Cafeteria Menu Ordering System

A cafeteria has a menu with the following items:

1. Tea - $2
2. Coffee - $3
3. Sandwich - $5
4. Burger - $7
5. Exit

Write a program to allow a customer to order items until they decide to exit. The program should:

1. Display the menu options.
2. Allow the user to select an item using a switch case.
3. Add the cost of the selected item to a total bill.
4. Use a loop to repeat until the user chooses the "Exit" option.
5. Include a "break" statement to exit the loop.
6. At the end, display the total bill.

Problem Requirements:

- Use a **`while`** or **`do-while`** loop to keep the menu running.
- Use a **`switch`** statement to handle the menu choices.
- Use a **`break`** to terminate the loop when the user selects "Exit."
- Use **`if`** statements to validate inputs if needed.

Scenario 2: Simple ATM Simulator

Write a program to simulate a simple ATM machine where:

1. The user starts with a fixed balance (e.g., $1000).
2. The program presents a menu with options:
   - 1: Check Balance
   - 2: Withdraw Money
   - 3: Deposit Money
   - 4: Exit
3. Based on the user's choice:
   - Display the current balance.
   - Allow withdrawal (if the withdrawal amount doesn't exceed the balance).
   - Allow deposit by adding the entered amount to the balance.
4. Exit the program when the user selects "Exit."

Problem Requirements:

- Use a **`do-while`** loop to keep displaying the menu until the user exits.
- Use a **`switch`** case for menu selection.
- Use **`if`** statements for balance checks and validation.
- Use a **`break`** statement to exit the menu loop.

Scenario 3: Odd or Even Counter

Write a program that counts how many odd and even numbers the user enters. The program should:

1. Continuously prompt the user to enter numbers.
2. Increment a counter for odd numbers or even numbers based on the input.
3. Stop when the user enters 0 and display the total count of odd and even numbers.
4. Ensure that 0 is not included in the count.

Problem Requirements:

- Use a **`while`** loop for repeated input.
- Use **`if`** conditions to check whether a number is odd or even.
- Use a **`break`** statement when the user enters 0.

Scenario 4: Simple Calculator

Write a program to create a simple calculator that:

1. Displays a menu to choose an operation:
    - 1: Addition
    - 2: Subtraction
    - 3: Multiplication
    - 4: Division
    - 5: Exit
2. Based on the selected option, prompts the user to enter two numbers.
3. Performs the chosen operation and displays the result.
4. Returns to the menu unless the user chooses to exit.

Problem Requirements:

- Use a **`do-while`** loop to display the menu repeatedly.
- Use a **`switch`** case to handle the selected operation.
- Use **`if`** conditions to validate inputs (e.g., avoid division by zero).
- Use a **`break`** statement to exit the loop when "Exit" is chosen.

Scenario 5: Password Checker

Write a program to validate a password. The program should:

1. Set a predefined password (e.g., "1234").
2. Allow the user three attempts to enter the correct password.
3. Display "Access granted" if the password is correct.
4. Display "Access denied" if all attempts are used up without success.
5. End the program after three attempts or a successful entry.

Problem Requirements:

- Use a **`for`** loop to allow up to three attempts.
- Use an **`if`** condition to check the password.
- Use a **`break`** statement to exit the loop when the password is correct.

Scenario 6: Grade Calculator

Write a program that calculates a student's grade based on their marks. The program should:

1. Prompt the user to enter their marks (0-100).
2. Use **`if-else`** conditions to assign a grade:
   - 90-100: A
   - 80-89: B
   - 70-79: C
   - 60-69: D
   - Below 60: F
3. Display the grade.
4. Use a loop to allow the user to calculate grades for multiple students until they choose to stop.

Problem Requirements:

- Use a **`do-while`** loop to repeat the grading process.
- Use **`if-else`** to determine the grade based on marks.
- Validate the input to ensure marks are between 0 and 100.

Scenario 7: Number Divisors

Write a program to find and display all divisors of a user-entered number. The program should:

1. Prompt the user to enter a positive integer.
2. Use a loop to find all numbers that divide evenly into the given number.
3. Display the divisors.

Example: If the input is **`12`**, the output should be **`1, 2, 3, 4, 6, 12.`**

Problem Requirements:

- Use a **`for`** loop to check each number from 1 to the entered number.
- Use an **`if`** statement to check for divisibility.

Scenario 8: Reverse Number Pattern

Write a program to display a reverse number pattern based on a user-entered number. The program should:

1. Prompt the user to enter a positive integer **n**.
2. Display a reverse triangle of numbers from **n** down to 1.

Example: For input **5**, the output should be:

```
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
```

Problem Requirements:

- Use nested `for` loops to generate the pattern.
- Validate input to ensure it is positive.

Scenario 9: Square and Cube Table

Write a program to generate a table of squares and cubes for numbers from 1 to **n**. The program should:

1. Prompt the user to enter a positive integer **n**.
2. Display a table with numbers, their squares, and cubes.

Example: For **`n = 3`**, the output should be:

```
Number        Square        Cube
  1             1             1
  2             4             8
  3             9             27
```

Problem Requirements:

- Use a `for` loop to calculate squares and cubes.
- Validate the input to ensure **n** is positive.

"Pull your code to Github account by creating new repository and share the link through google classroom"