Review Material 01:

Program Simulation using C#

Overview

Program simulation involves modeling the behavior of a real-world process or system using computer code. In C#, simulation is often done through logic-based structures, data handling, loops, and control statements to mimic scenarios such as banking systems, queue management, vending machines, and more.

Core Concepts

1. **Input and Output (I/O)**
2. **Control Structures (if-else, switch)**
3. **Loops (for, while, do-while)**
4. **Arrays and Lists**
5. **Classes and Objects**
6. **File Handling (Optional in simulations)**
7. **Randomization**
8. **State Management**

Example 1: ATM Simulation

```
using System;

class ATMSimulator {
    static void Main() {
        int balance = 10000;
        int pin = 1234;
        Console.Write("Enter PIN: ");
        int inputPin = Convert.ToInt32(Console.ReadLine());

        if (inputPin == pin) {
            Console.WriteLine("1. Check Balance\n2. Deposit\n3. Withdraw\n4.
Exit");
            Console.Write("Choose option: ");
            int choice = Convert.ToInt32(Console.ReadLine());

            switch (choice) {
                case 1:
                    Console.WriteLine("Current Balance: " + balance);
                    break;
                case 2:
                    Console.Write("Enter amount to deposit: ");
                    int deposit = Convert.ToInt32(Console.ReadLine());
                    balance += deposit;
                    Console.WriteLine("New Balance: " + balance);
                    break;
                case 3:
                    Console.Write("Enter amount to withdraw: ");
                    int withdraw = Convert.ToInt32(Console.ReadLine());
                    if (withdraw <= balance) {
                        balance -= withdraw;
                        Console.WriteLine("Withdrawn. New Balance: " + balance);
                    } else {
```

```
                        Console.WriteLine("Insufficient balance.");
                    }
                    break;
                case 4:
                    Console.WriteLine("Thank you!");
                    break;
                default:
                    Console.WriteLine("Invalid choice.");
                    break;
            }
        } else {
            Console.WriteLine("Incorrect PIN.");
        }
    }
}
```

Run Code: https://www.programiz.com/online-compiler/7fkzHLNASyLNF

✅ Explanation:

- Simulates basic ATM operations like deposit, withdrawal, and balance inquiry.
- Uses `switch` and `if-else` control structures for decision-making.
- Demonstrates input handling and basic arithmetic operations.

💡 Example 2: Dice Roll Simulator

```
using System;

class DiceSimulator {
    static void Main() {
        Random rand = new Random();
        Console.WriteLine("Press Enter to roll the dice...");
        Console.ReadLine();

        int roll = rand.Next(1, 7);
        Console.WriteLine("You rolled a: " + roll);
    }
}
```

Run Code: https://www.programiz.com/online-compiler/0hIy6oZfpqb8O

✅ Explanation:

- Uses `Random` class to simulate rolling a six-sided dice.
- Demonstrates basic random number generation.
- Ideal for learning input/output and randomization.

💡 Example 3: Vending Machine Simulator

```
using System;

class VendingMachine {
    static void Main() {
        string[] items = { "Soda", "Chips", "Candy" };
        int[] prices = { 25, 15, 10 };

        Console.WriteLine("Available Items:");
```

```
        for (int i = 0; i < items.Length; i++) {
            Console.WriteLine($"{i + 1}. {items[i]} - {prices[i]} PHP");
        }

        Console.Write("Insert amount (PHP): ");
        int money = Convert.ToInt32(Console.ReadLine());

        Console.Write("Select item number: ");
        int choice = Convert.ToInt32(Console.ReadLine()) - 1;

        if (choice >= 0 && choice < items.Length) {
            if (money >= prices[choice]) {
                Console.WriteLine($"Dispensing {items[choice]}...");
                Console.WriteLine($"Change: {money - prices[choice]} PHP");
            } else {
                Console.WriteLine("Insufficient funds.");
            }
        } else {
            Console.WriteLine("Invalid selection.");
        }
    }
}
```

Run Code: https://www.programiz.com/online-compiler/3BTGIbKJQFTpW

✅ Explanation:

- Simulates a vending machine interface with item selection and change calculation.
- Introduces arrays and `for` loops to manage items and prices.

💡 Example 4: Queue Simulation (Ticketing System)

```
using System;
using System.Collections.Generic;

class QueueSimulator {
    static void Main() {
        Queue<string> queue = new Queue<string>();

        while (true) {
            Console.WriteLine("\n1. Enqueue\n2. Dequeue\n3. View Queue\n4. Exit");
            Console.Write("Choose: ");
            int choice = Convert.ToInt32(Console.ReadLine());

            switch (choice) {
                case 1:
                    Console.Write("Enter name: ");
                    string name = Console.ReadLine();
                    queue.Enqueue(name);
                    Console.WriteLine($"{name} added to queue.");
                    break;
                case 2:
                    if (queue.Count > 0) {
                        Console.WriteLine($"{queue.Dequeue()} removed from
queue.");
                    } else {
                        Console.WriteLine("Queue is empty.");
                    }
                    break;
                case 3:
```

```
                    Console.WriteLine("Current Queue:");
                    foreach (var person in queue)
                        Console.WriteLine(person);
                    break;
                case 4:
                    return;
                default:
                    Console.WriteLine("Invalid choice.");
                    break;
            }
        }
    }
}
```

Run Code: https://www.programiz.com/online-compiler/43aTPlAZCJqvC

✅ Explanation:

- Simulates a queue (FIFO) using `Queue<T>` collection.
- Useful in customer service simulations like ticketing or bank lines.

💡 Example 5: Basic Traffic Light Simulation

```
using System;
using System.Threading;

class TrafficLightSimulator {
    static void Main() {
        string[] lights = { "Green", "Yellow", "Red" };
        int[] durations = { 3000, 2000, 5000 }; // milliseconds

        for (int i = 0; i < lights.Length; i++) {
            Console.WriteLine(lights[i] + " light");
            Thread.Sleep(durations[i]);
        }

        Console.WriteLine("Cycle complete.");
    }
}
```

Run Code: https://www.programiz.com/online-compiler/3ABEhqEYusN93

✅ Explanation:

- Uses `Thread.Sleep()` to simulate timed light changes.
- Demonstrates sequence control and timed events in simulations.

Summary of Key Points

Concept	Description
`Console.ReadLine()`	Gets user input from the console
`Convert.ToInt32()`	Converts string input to integer
`Random.Next()`	Generates random numbers
`switch-case`	Control flow for multiple options
`Queue<T>`	First-In-First-Out collection
`Array`	Stores multiple items of the same type
`Thread.Sleep()`	Delays program execution

Review MCQ Questions

Q1. Which method is used to generate random numbers in C#?
A.) Math.Random()
B.) Random.Next()
C.) Generate.Random()
D.) Random.Get()

Q2. What collection class implements a FIFO structure?
A.) Stack
B.) List
C.) Queue
D.) Dictionary

Q3. What does `Thread.Sleep()` do in C#?
A.) Ends the program
B.) Pauses execution
C.) Repeats a loop
D.) Generates a random delay

Q4. In a vending machine simulation, which structure best stores item names and prices?
A.) Dictionary
B.) Array
C.) Stack
D.) Queue

Q5. Which of the following is NOT a valid control structure in C#?
A.) if-else
B.) select
C.) switch
D.) while

Q6. What is the purpose of `Console.ReadLine()`?
A.) To display output
B.) To clear the screen
C.) To read input
D.) To terminate the program

Q7. What happens if you call `Queue.Dequeue()` on an empty queue?
A.) Returns null
B.) Removes the last item
C.) Throws an exception
D.) Does nothing

Q8. Which operator is used to combine strings in C#?
A.) +
B.) &
C.) &&
D.) ++

Q9. What value is returned by `Random.Next(1, 7)`?
A.) Always 6
B.) 1 to 7 inclusive
C.) 0 to 6 inclusive
D.) 1 to 6 inclusive

Q10. What is the correct way to terminate a program in a `switch` case?
A.) break
B.) exit
C.) stop
D.) return

Answers: https://github.com/clydeatmcm/IT104/blob/main/ICT-Proficiency-Test/CSharp/01-AnswerKeys.txt