

C# ASSIGNMENT 1

Task 1

In a bank, you have been given the task is to create a program that checks if a customer is eligible for a loan based on their credit score and income. The eligibility criteria are as follows:

- Credit Score must be above 700.
- Annual Income must be at least \$50,000.

Tasks:

1. Write a program that takes the customer's credit score and annual income as input.
2. Use conditional statements (if-else) to determine if the customer is eligible for a loan.
3. Display an appropriate message based on eligibility.

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Assignment
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter the Credit Score: ");
            int creditscore = int.Parse(Console.ReadLine());

            Console.WriteLine("Enter the Annual Income");
            int annualIncome = int.Parse(Console.ReadLine());

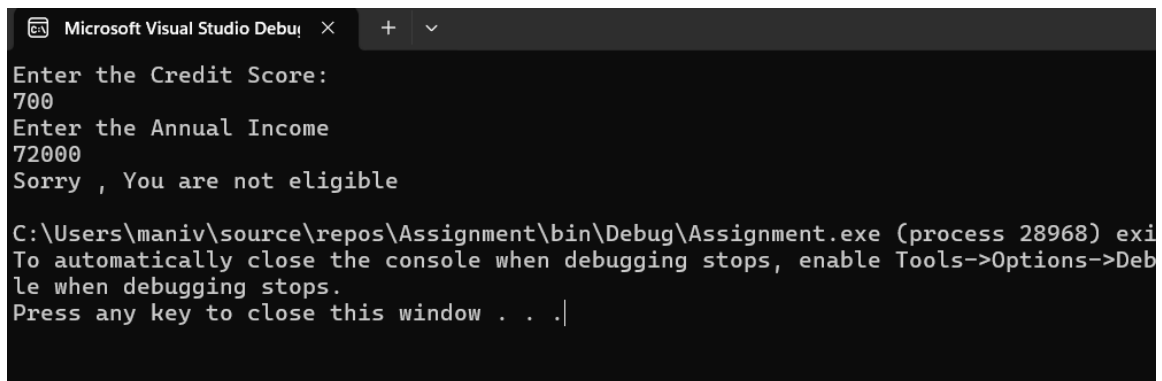
            if (creditscore > 700 && annualIncome >= 50000)
            {
                Console.WriteLine("You're Eligible for a loan");
            }
        }
    }
}
```

```

        else
        {
            Console.WriteLine("Sorry , You are not eligible");
        }
    }
}
}
}

```

Output:



The screenshot shows a Visual Studio Debug Console window with a dark background. The text inside the console is as follows:

```

Enter the Credit Score:
700
Enter the Annual Income
72000
Sorry , You are not eligible

C:\Users\maniv\source\repos\Assignment\bin\Debug\Assignment.exe (process 28968) exited
To automatically close the console when debugging stops, enable Tools->Options->Debu
le when debugging stops.
Press any key to close this window . . .|

```

Task 2

Create a program that simulates an ATM transaction. Display options such as "Check Balance," "Withdraw," "Deposit,". Ask the user to enter their current balance and the amount they want to withdraw or deposit. Implement checks to ensure that the withdrawal amount is not greater than the available balance and that the withdrawal amount is in multiples of 100 or 500. Display appropriate messages for success or failure.

Code:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace Assignment
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("enter the current balance");

            double balance = double.Parse(Console.ReadLine());
            Console.WriteLine("\n ATM MENU");
            Console.WriteLine("1.Check balance");
            Console.WriteLine("2.withdraw");
            Console.WriteLine("3.Deposit");
            Console.WriteLine("4.exit");

            Console.WriteLine("select ur Choice");

            int choice = int.Parse(Console.ReadLine());
            switch (choice)
            {
                case 1:
                    Console.WriteLine("your balance is" + balance);
                    break;

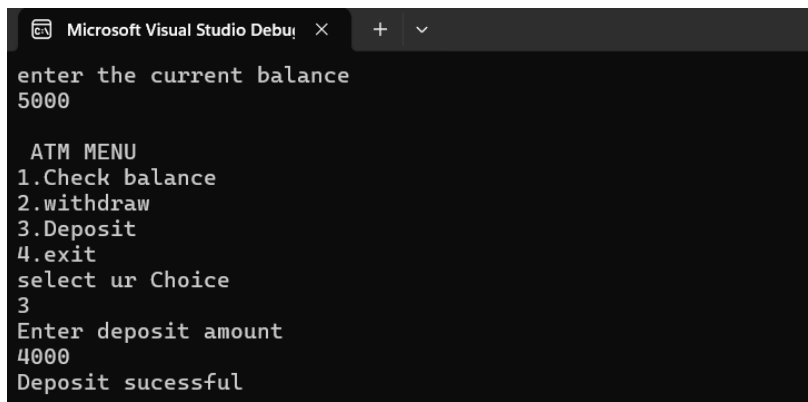
                case 2:
                    Console.WriteLine("Enter amount to withdraw:");
                    double withdraw = double.Parse(Console.ReadLine());
                    if (withdraw > balance)
                    {
                        Console.WriteLine("insufficient balance");
                    }
                    else if (withdraw % 100 != 0 && withdraw % 500 != 0)

```

```

    {
        Console.WriteLine("please enter amount in multiples of 100 or 500");
    }
    else
    {
        balance -= withdraw;
        Console.WriteLine("transaction sucessful");
    }
    break;
case 3:
    Console.WriteLine("Enter deposit amount");
    double deposit = double.Parse(Console.ReadLine());
    balance += deposit;
    Console.WriteLine("Deposit sucessful");
    break;
case 4:
    Console.WriteLine("thank you using ATM");
    break;
default:
    Console.WriteLine("Invalid choice");
    break;
}
}
}
}

```



```

Microsoft Visual Studio Debug Console
enter the current balance
5000

ATM MENU
1.Check balance
2.withdraw
3.Deposit
4.exit
select ur Choice
3
Enter deposit amount
4000
Deposit sucessful

```

Task 3: Loop Structures

You are responsible for calculating compound interest on savings accounts for bank customers. You need to calculate the future balance for each customer's savings account after a certain number of years.

Tasks:

1. Create a program that calculates the future balance of a savings account.
2. Use a loop structure (e.g., for loop) to calculate the balance for multiple customers.
3. Prompt the user to enter the initial balance, annual interest rate, and the number of years.
4. Calculate the future balance using the formula:
$$\text{future_balance} = \text{initial_balance} * (1 + \text{annual_interest_rate}/100)^{\text{years}}$$
5. Display the future balance for each customer.

Code:

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("enter the number of customers");

            int customers = int.Parse(Console.ReadLine());

            for (int i = 0; i < customers; i++) {

                Console.WriteLine($"Customer {i + 1}");


                Console.WriteLine("Enter initial balance:");

                double balance = double.Parse(Console.ReadLine());

                Console.WriteLine("Enter Annual interest rate");

                double airate = double.Parse(Console.ReadLine());

                Console.WriteLine("enter the number of years");

                int years = int.Parse(Console.ReadLine());

                double futureBalance = balance * Math.Pow((1 + airate / 100), years);

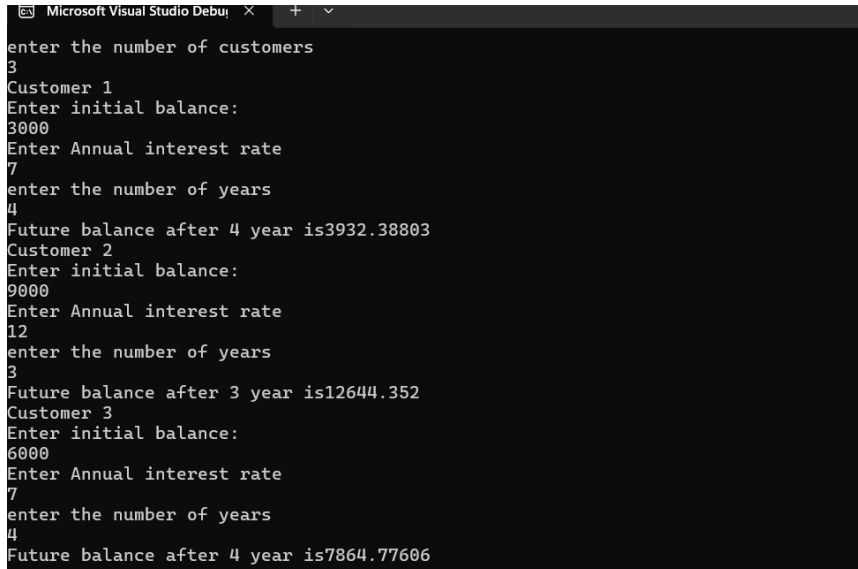
                Console.WriteLine($"Future balance after {years} year is" + futureBalance);
            }
        }
    }
}
```

```

    }
}
}
}

```

Output:



```

Microsoft Visual Studio Debug Console
enter the number of customers
3
Customer 1
Enter initial balance:
3000
Enter Annual interest rate
7
enter the number of years
4
Future balance after 4 year is3932.38803
Customer 2
Enter initial balance:
9000
Enter Annual interest rate
12
enter the number of years
3
Future balance after 3 year is12644.352
Customer 3
Enter initial balance:
6000
Enter Annual interest rate
7
enter the number of years
4
Future balance after 4 year is7864.77606

```

Task 4: Looping, Array and Data Validation

You are tasked with creating a program that allows bank customers to check their account balances.

The program should handle multiple customer accounts, and the customer should be able to enter their account number, balance to check the balance. Account Number should be in the format (first four letters should be INDB followed by 4 numbers , Eg INDB2345)

Tasks:

1. Create a C# program that simulates a bank with multiple customer accounts.
2. Use a loop (e.g., while loop) to repeatedly ask the user for their account number and balance until they enter a valid account number.
3. Validate the account number entered by the user.
4. If the account number is valid, display the account balance. If not, ask the user to try again.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace Assignment
{
    internal class Program
    {
        static void Main(string[] args)
        {
            string[][] accounts = new string[][]
        {
            new string[] { "INDB1234", "5000.75" },
            new string[] { "INDB5678", "12000.50" },
            new string[] { "INDB9101", "750.25" }
        };

            Console.WriteLine("Welcome to the Bank System!");

            while (true)
            {
                Console.Write("Enter your account number (Format: INDB1234): ");
                string accountNumber = Console.ReadLine();
                Console.WriteLine(accountNumber.Substring(4));

                if (!(accountNumber.Length == 8 &&
                    accountNumber.StartsWith("INDB") &&
                    int.TryParse(accountNumber.Substring(4), out _)))
                {
                    Console.WriteLine("Invalid account number format! Try again.");
                    continue;
                }
                bool found = false;
                foreach (var account in accounts)
                {
                    if (account[0] == accountNumber)
                    {
                        Console.WriteLine($"Your account balance:" + account[1]);
                    }
                }
            }
        }
    }
}

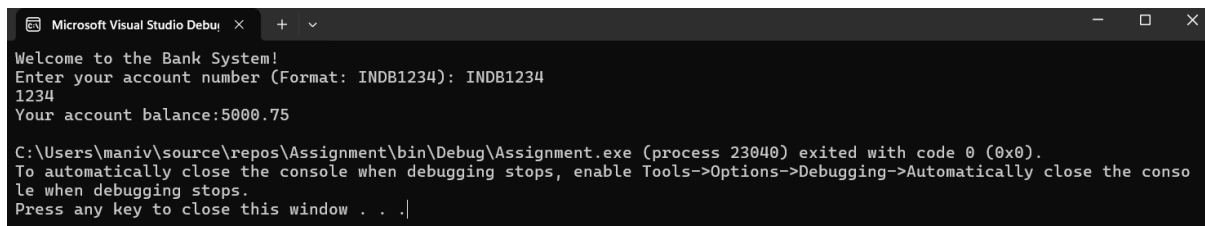
```

```

        found = true;
        break;
    }
}
if (!found)
{
    Console.WriteLine("Account not found! Try again.");
}
else
{
    break;
}
}
}
}
}

```

Output:



```

Microsoft Visual Studio Debug Console
Welcome to the Bank System!
Enter your account number (Format: INDB1234): INDB1234
1234
Your account balance:5000.75
C:\Users\maniv\source\repos\Assignment\bin\Debug\Assignment.exe (process 23040) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .|

```

Task 5: Password Validation

Write a program that prompts the user to create a password for their bank account. Implement if conditions to validate the password according to these rules:

- The password must be at least 8 characters long.
- It must contain at least one uppercase letter.
- It must contain at least one digit.
- Display appropriate messages to indicate whether their password is valid or not.

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Assignment
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter your password");
            String password = Console.ReadLine();

            if (IsValidPass(password))
            {
                Console.WriteLine("the password is valid");
            }

            else
            {
                Console.WriteLine("The password is invalid");
            }
        }

        static bool IsValidPass(String password)
        {
            if (password.Length < 8)
            {
```

```
        Console.WriteLine("The password must contain 8 letter");
        return false;
    }

    bool upper = false;
    bool digit = false;

    foreach (char c in password)
    {
        if (char.IsUpper(c))
        {
            upper = true;
        }

        if (char.IsDigit(c))
        {
            digit = true;
        }
    }

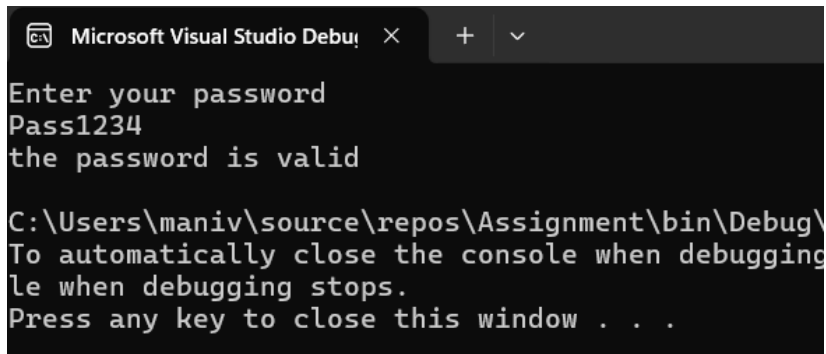
    if (!upper)
    {
        Console.WriteLine("Password must contain one uppercase");
        return false;
    }

    if (!digit)
    {
        Console.WriteLine("Password must contain one digit");
        return false;
    }

    return true;
}
}
```

```
}
```

Output:

A screenshot of a Microsoft Visual Studio Debug Console window. The window has a title bar with the Visual Studio logo, the text 'Microsoft Visual Studio Debug Console', and standard window controls. The console output is as follows:

```
Enter your password
Pass1234
the password is valid

C:\Users\maniv\source\repos\Assignment\bin\Debug\
To automatically close the console when debugging
le when debugging stops.
Press any key to close this window . . .
```

Task 6: Password Validation

Create a program that maintains a list of bank transactions (deposits and withdrawals) for a customer. Use a while loop to allow the user to keep adding transactions until they choose to exit. Display the transaction history upon exit using looping statements.

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Assignment
{
    internal class Program
    {
        static void Main(string[] args)
        {
            List<string> transactions = new List<string>();
            double balance = 0;
            bool running = true;

            Console.WriteLine("Welcome to Bank Transactions!");

            while (running)
            {
```

```

Console.WriteLine("Choose an option:");
Console.WriteLine("1. Deposit");
Console.WriteLine("2. Withdraw");
Console.WriteLine("3. Exit & Show History");
Console.Write("Enter your choice: ");

string choice = Console.ReadLine();

switch (choice)
{
    case "1":
        Console.Write("Enter deposit amount: ");
        double depositAmount = double.Parse(Console.ReadLine());
        if (depositAmount > 0)
        {
            balance += depositAmount;
            transactions.Add($"Deposited: {depositAmount} | New Balance: {balance}");
            Console.WriteLine($"Deposited {depositAmount}. New Balance: {balance}");
        }
        else
        {
            Console.WriteLine("Invalid deposit amount. Please enter a valid number.");
        }
        break;

    case "2":
        Console.Write("Enter withdrawal amount: ");
        double withdrawAmount = double.Parse(Console.ReadLine());
        if (withdrawAmount > 0)
        {
            if (withdrawAmount > balance)
            {
                Console.WriteLine("Insufficient balance");
            }
        }
    }
}

```

```

        else
        {
            balance -= withdrawAmount;
            transactions.Add($"Withdrew:{ withdrawAmount} | New Balance: {balance}");
            Console.WriteLine($"Withdrawn {withdrawAmount}. New Balance: {balance}");
        }
    }
    else
    {
        Console.WriteLine("Invalid withdrawal amount");
    }
    break;

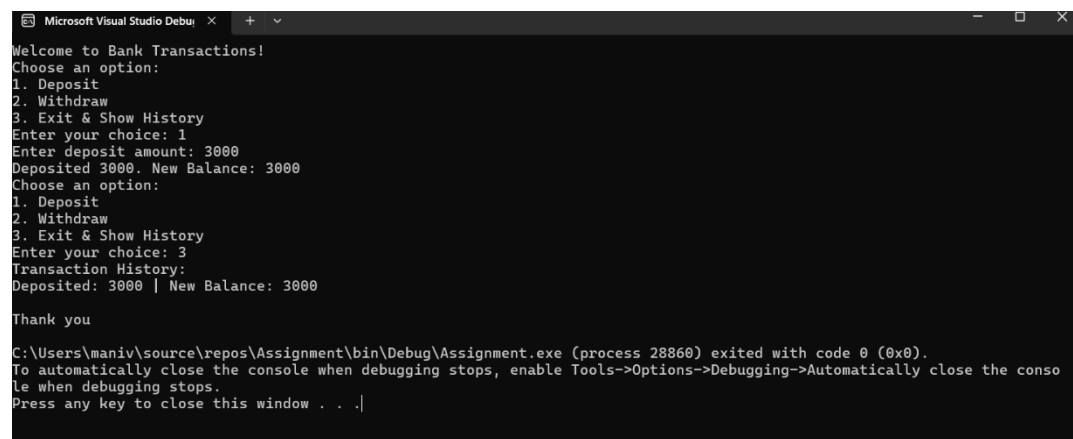
case "3":
    running = false;
    break;

default:
    Console.WriteLine("Invalid choice! Please enter 1, 2, or 3.");
    break;
}
}
Console.WriteLine("Transaction History:");
if (transactions.Count == 0)
{
    Console.WriteLine("No transactions done");
}
else
{
    foreach (string transaction in transactions)
    {
        Console.WriteLine(transaction);
    }
}
}

```

```
        Console.WriteLine("\nThank you ");  
    }  
}  
}
```

Output:



```
Microsoft Visual Studio Debug Console  
Welcome to Bank Transactions!  
Choose an option:  
1. Deposit  
2. Withdraw  
3. Exit & Show History  
Enter your choice: 1  
Enter deposit amount: 3000  
Deposited 3000. New Balance: 3000  
Choose an option:  
1. Deposit  
2. Withdraw  
3. Exit & Show History  
Enter your choice: 3  
Transaction History:  
Deposited: 3000 | New Balance: 3000  
  
Thank you  
  
C:\Users\maniv\source\repos\Assignment\bin\Debug\Assignment.exe (process 28860) exited with code 0 (0x0).  
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.  
Press any key to close this window . . .|
```