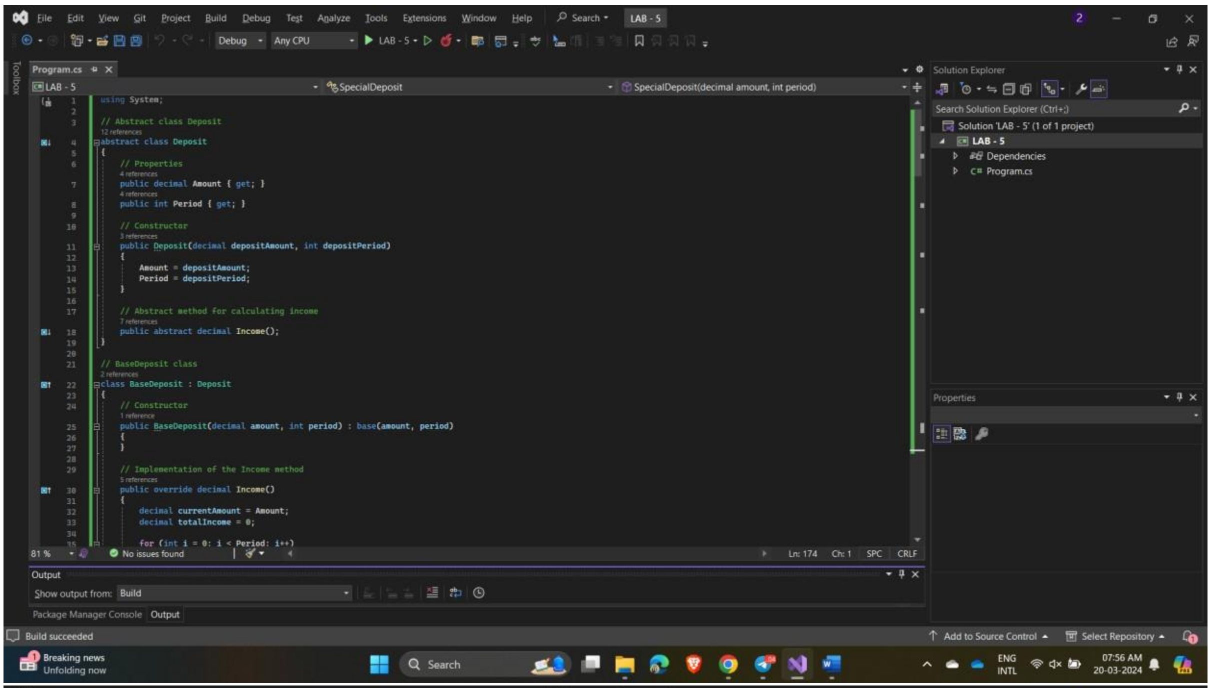
2100030522

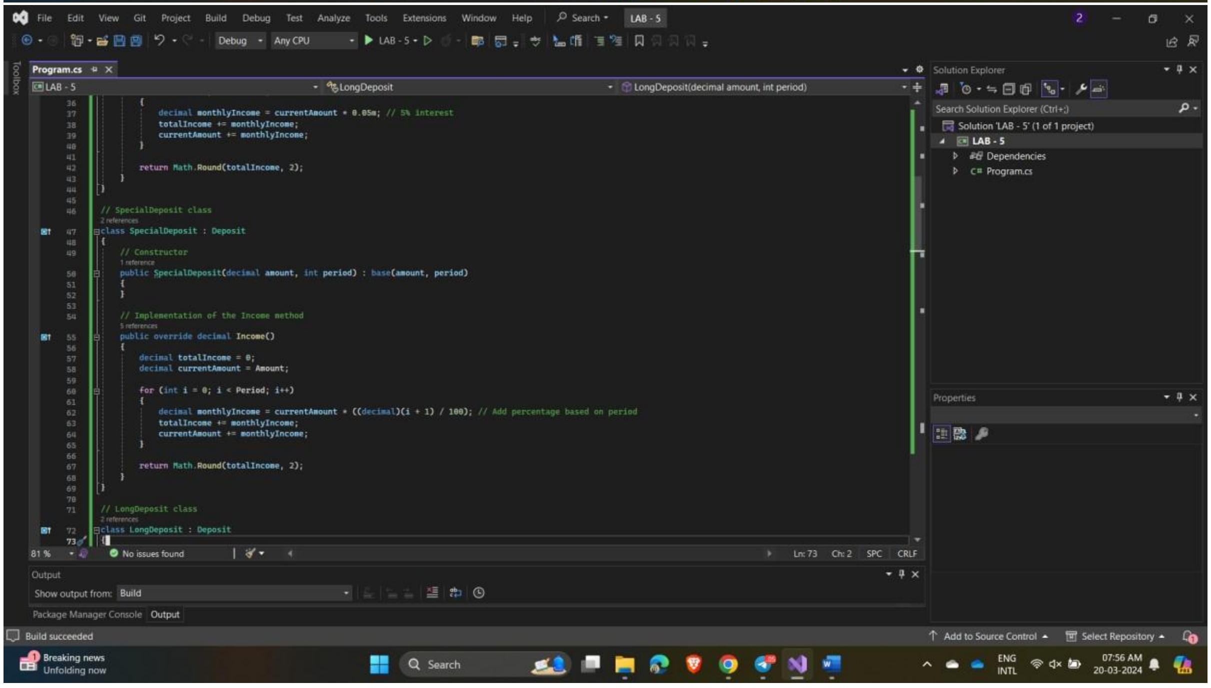
Syed Hasif Alisha

LAB - 5.1



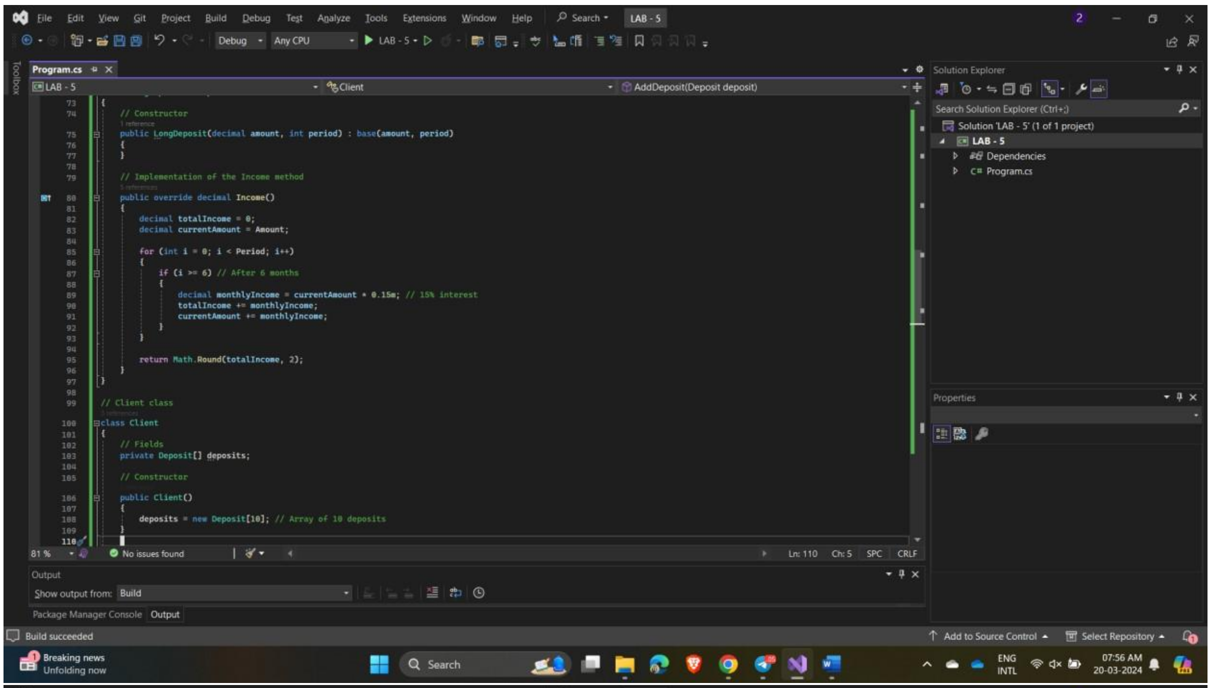
```csharp
using System;

// Abstract class Deposit
abstract class Deposit
{
    // Properties
    public decimal Amount { get; }
    public int Period { get; }

    // Constructor
    public Deposit(decimal depositAmount, int depositPeriod)
    {
        Amount = depositAmount;
        Period = depositPeriod;
    }

    // Abstract method for calculating income
    public abstract decimal Income();
}

// BaseDeposit class
class BaseDeposit : Deposit
{
    // Constructor
    public BaseDeposit(decimal amount, int period) : base(amount, period)
    {
    }

    // Implementation of the Income method
    public override decimal Income()
    {
        decimal currentAmount = Amount;
        decimal totalIncome = 0;

        for (int i = 0; i < Period; i++)
```

Build succeeded

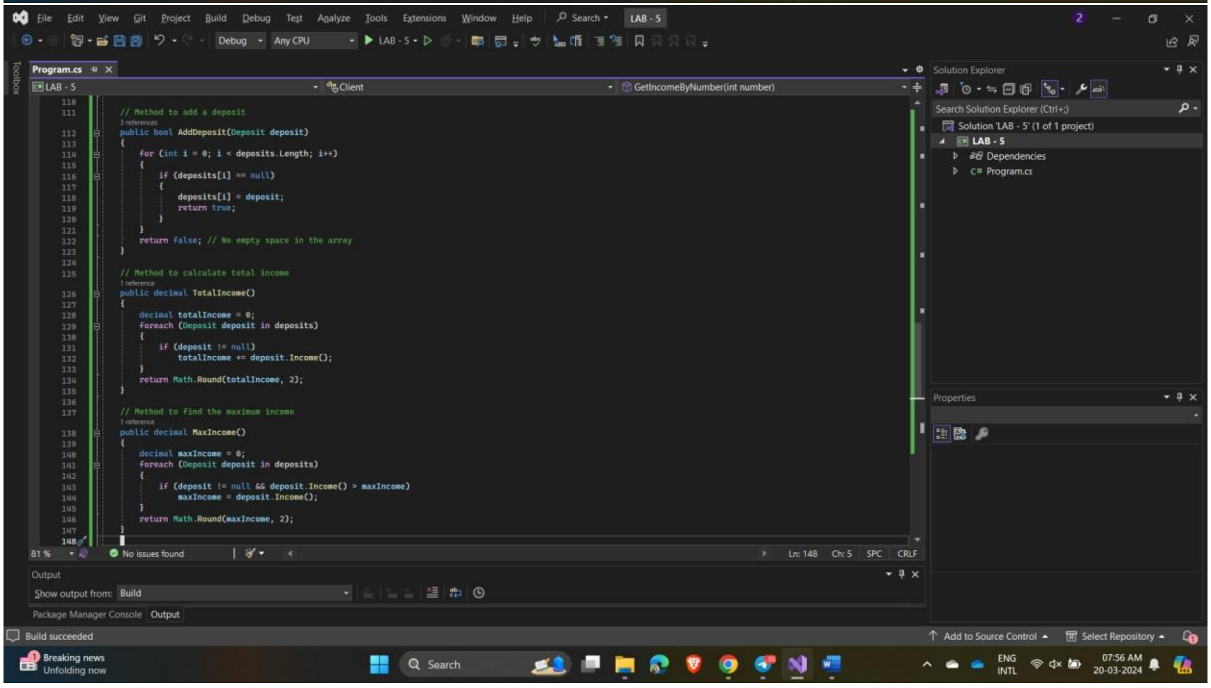


```csharp
        {
            decimal monthlyIncome = currentAmount * 0.05m; // 5% interest
            totalIncome += monthlyIncome;
            currentAmount += monthlyIncome;
        }

        return Math.Round(totalIncome, 2);
    }
}

// SpecialDeposit class
class SpecialDeposit : Deposit
{
    // Constructor
    public SpecialDeposit(decimal amount, int period) : base(amount, period)
    {
    }

    // Implementation of the Income method
    public override decimal Income()
    {
        decimal totalIncome = 0;
        decimal currentAmount = Amount;

        for (int i = 0; i < Period; i++)
        {
            decimal monthlyIncome = currentAmount * ((decimal)(i + 1) / 100); // Add percentage based on period
            totalIncome += monthlyIncome;
            currentAmount += monthlyIncome;
        }

        return Math.Round(totalIncome, 2);
    }
}

// LongDeposit class
class LongDeposit : Deposit
{
```

Build succeeded

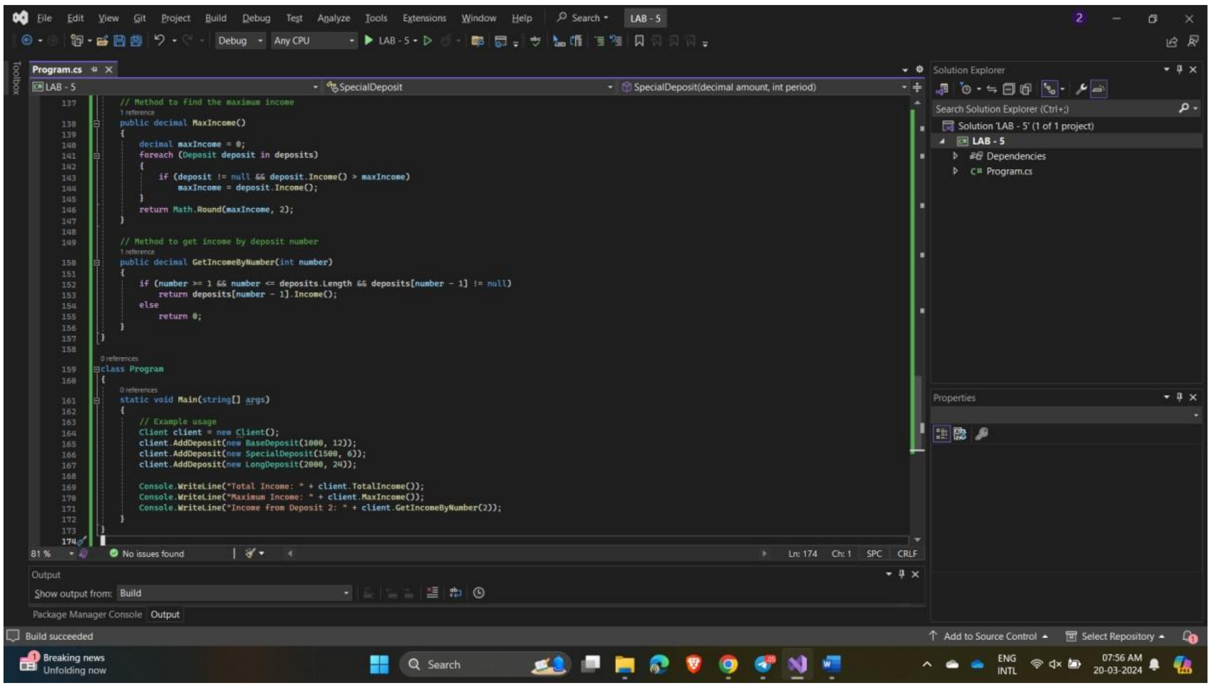


```
    {
        // Constructor
        public LongDeposit(decimal amount, int period) : base(amount, period)
        {
        }

        // Implementation of the Income method
        public override decimal Income()
        {
            decimal totalIncome = 0;
            decimal currentAmount = Amount;

            for (int i = 0; i < Period; i++)
            {
                if (i >= 6) // After 6 months
                {
                    decimal monthlyIncome = currentAmount * 0.15m; // 15% interest
                    totalIncome += monthlyIncome;
                    currentAmount += monthlyIncome;
                }
            }

            return Math.Round(totalIncome, 2);
        }
    }

    // Client class
    class Client
    {
        // Fields
        private Deposit[] deposits;

        // Constructor

        public Client()
        {
            deposits = new Deposit[10]; // Array of 10 deposits
        }
```



```

        // Method to add a deposit
        public bool AddDeposit(Deposit deposit)
        {
            for (int i = 0; i < deposits.Length; i++)
            {
                if (deposits[i] == null)
                {
                    deposits[i] = deposit;
                    return true;
                }
            }
            return false; // No empty space in the array
        }

        // Method to calculate total income
        public decimal TotalIncome()
        {
            decimal totalIncome = 0;
            foreach (Deposit deposit in deposits)
            {
                if (deposit != null)
                    totalIncome += deposit.Income();
            }
            return Math.Round(totalIncome, 2);
        }

        // Method to find the maximum income
        public decimal MaxIncome()
        {
            decimal maxIncome = 0;
            foreach (Deposit deposit in deposits)
            {
                if (deposit != null && deposit.Income() > maxIncome)
                    maxIncome = deposit.Income();
            }
            return Math.Round(maxIncome, 2);
        }
```

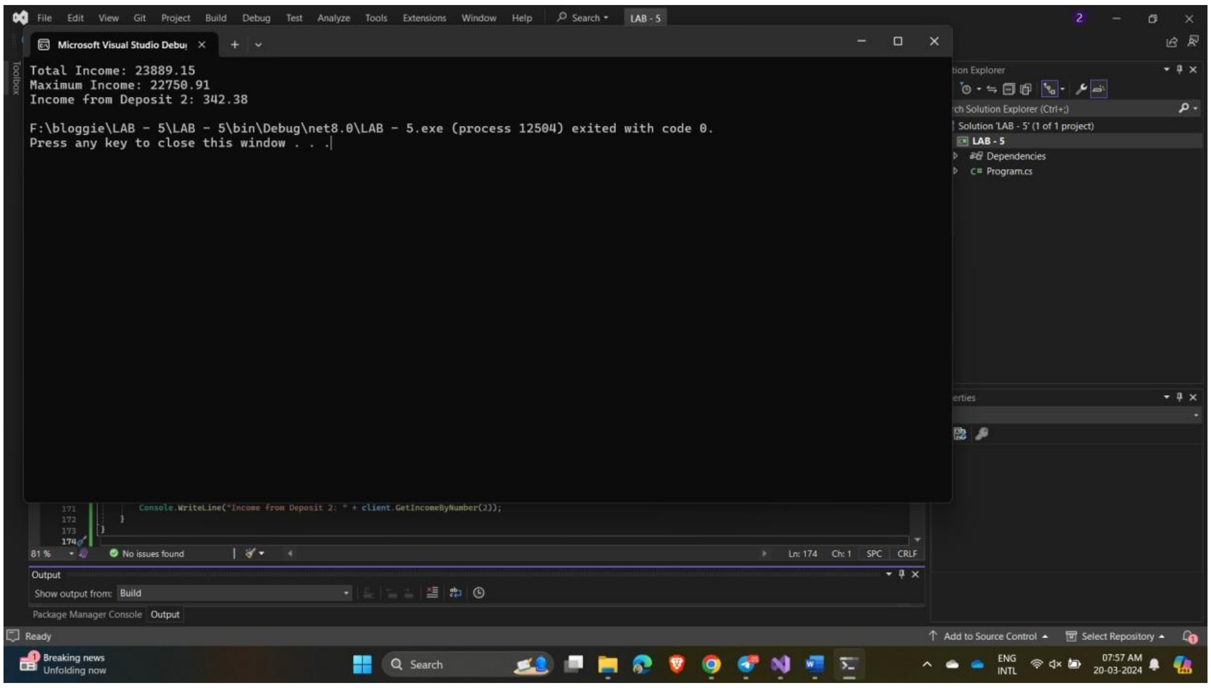
Build succeeded



```
        // Method to find the maximum income
        public decimal MaxIncome()
        {
            decimal maxIncome = 0;
            foreach (Deposit deposit in deposits)
            {
                if (deposit != null && deposit.Income() > maxIncome)
                    maxIncome = deposit.Income();
            }
            return Math.Round(maxIncome, 2);
        }

        // Method to get income by deposit number
        public decimal GetIncomeByNumber(int number)
        {
            if (number >= 1 && number <= deposits.Length && deposits[number - 1] != null)
                return deposits[number - 1].Income();
            else
                return 0;
        }
    }

class Program
{
    static void Main(string[] args)
    {
        // Example usage
        Client client = new Client();
        client.AddDeposit(new BaseDeposit(1000, 12));
        client.AddDeposit(new SpecialDeposit(1500, 6));
        client.AddDeposit(new LongDeposit(2000, 24));

        Console.WriteLine("Total Income: " + client.TotalIncome());
        Console.WriteLine("Maximum Income: " + client.MaxIncome());
        Console.WriteLine("Income from Deposit 2: " + client.GetIncomeByNumber(2));
    }
}
```

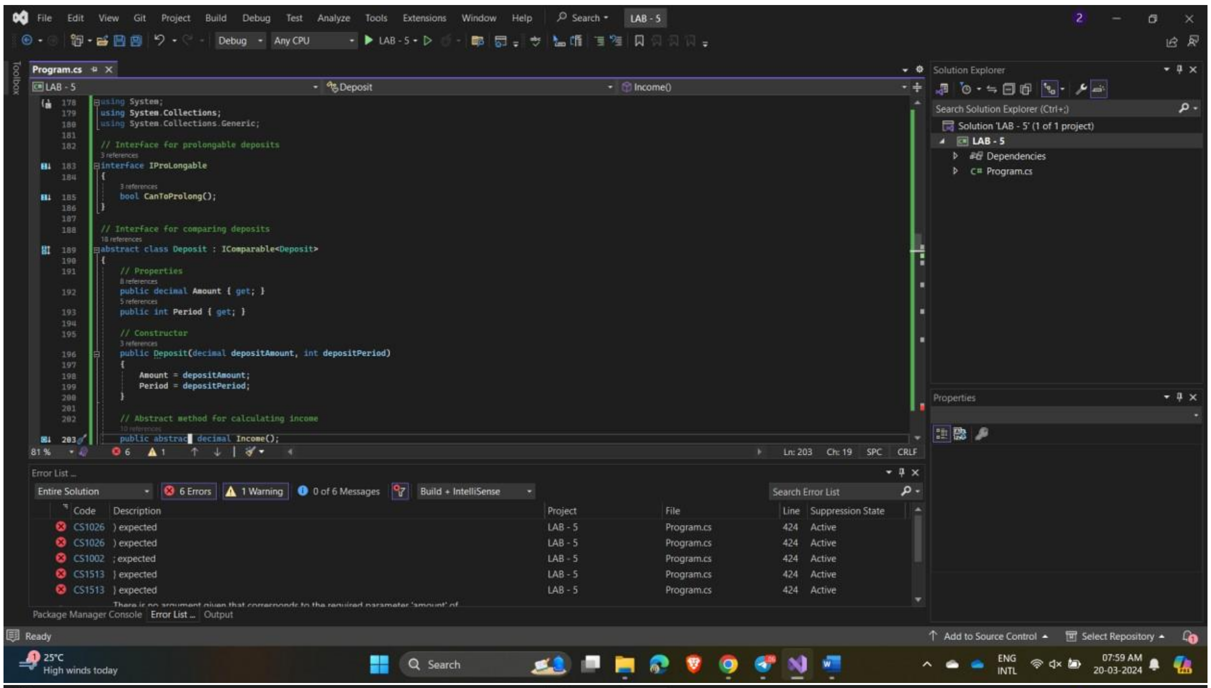
Build succeeded

OUTPUT :



```
Total Income: 23889.15
Maximum Income: 22750.91
Income from Deposit 2: 342.38

F:\bloggie\LAB - 5\LAB - 5\bin\Debug\net8.0\LAB - 5.exe (process 12504) exited with code 0.
Press any key to close this window . . .
```

LAB – 5.2



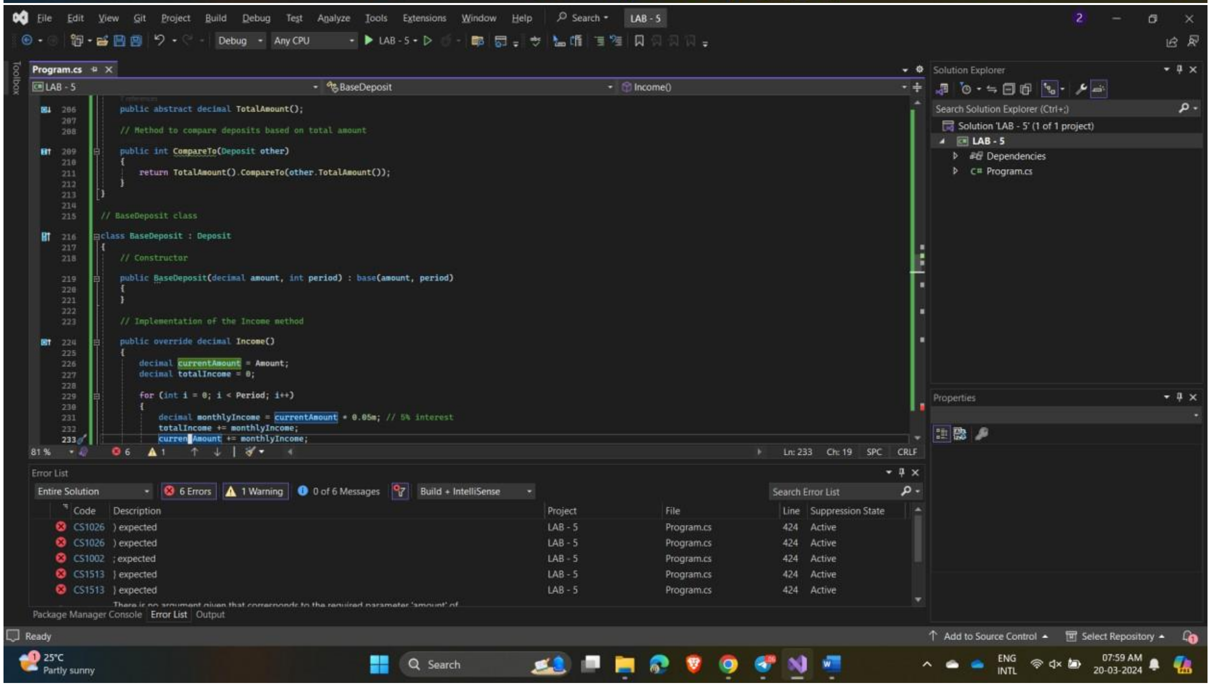
```csharp
using System;
using System.Collections;
using System.Collections.Generic;

// Interface for prolongable deposits
interface IProLongable
{
    bool CanToProlong();
}

// Interface for comparing deposits
abstract class Deposit : IComparable<Deposit>
{
    // Properties
    public decimal Amount { get; }
    public int Period { get; }

    // Constructor
    public Deposit(decimal depositAmount, int depositPeriod)
    {
        Amount = depositAmount;
        Period = depositPeriod;
    }

    // Abstract method for calculating income
    public abstract decimal Income();
```

Error List

Entire Solution | 6 Errors | 1 Warning | 0 of 6 Messages | Build + IntelliSense

Code	Description	Project	File	Line	Suppression State
CS1026	) expected	LAB - 5	Program.cs	424	Active
CS1026	) expected	LAB - 5	Program.cs	424	Active
CS1002	; expected	LAB - 5	Program.cs	424	Active
CS1513	} expected	LAB - 5	Program.cs	424	Active
CS1513	} expected	LAB - 5	Program.cs	424	Active
	There is no argument given that corresponds to the required parameter 'amount' of				

Ready



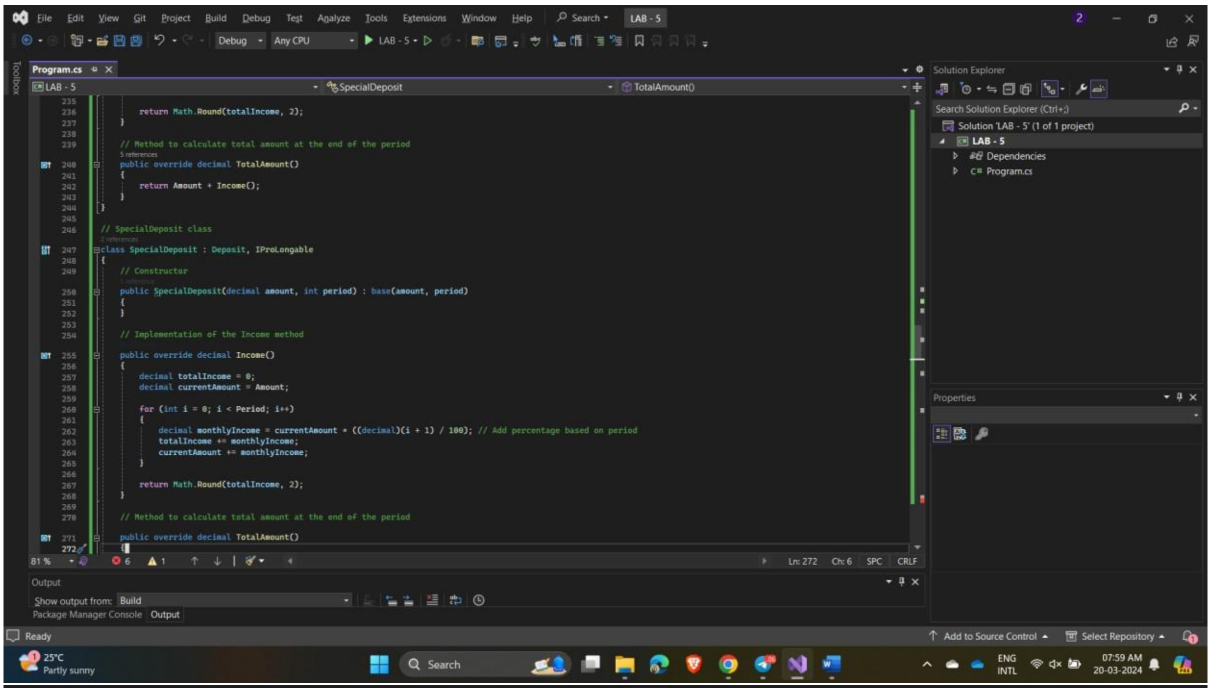
```csharp
    public abstract decimal TotalAmount();

    // Method to compare deposits based on total amount

    public int CompareTo(Deposit other)
    {
        return TotalAmount().CompareTo(other.TotalAmount());
    }
}

// BaseDeposit class

class BaseDeposit : Deposit
{
    // Constructor

    public BaseDeposit(decimal amount, int period) : base(amount, period)
    {
    }

    // Implementation of the Income method

    public override decimal Income()
    {
        decimal currentAmount = Amount;
        decimal totalIncome = 0;

        for (int i = 0; i < Period; i++)
        {
            decimal monthlyIncome = currentAmount * 0.05m; // 5% interest
            totalIncome += monthlyIncome;
            currentAmount += monthlyIncome;
```

Error List

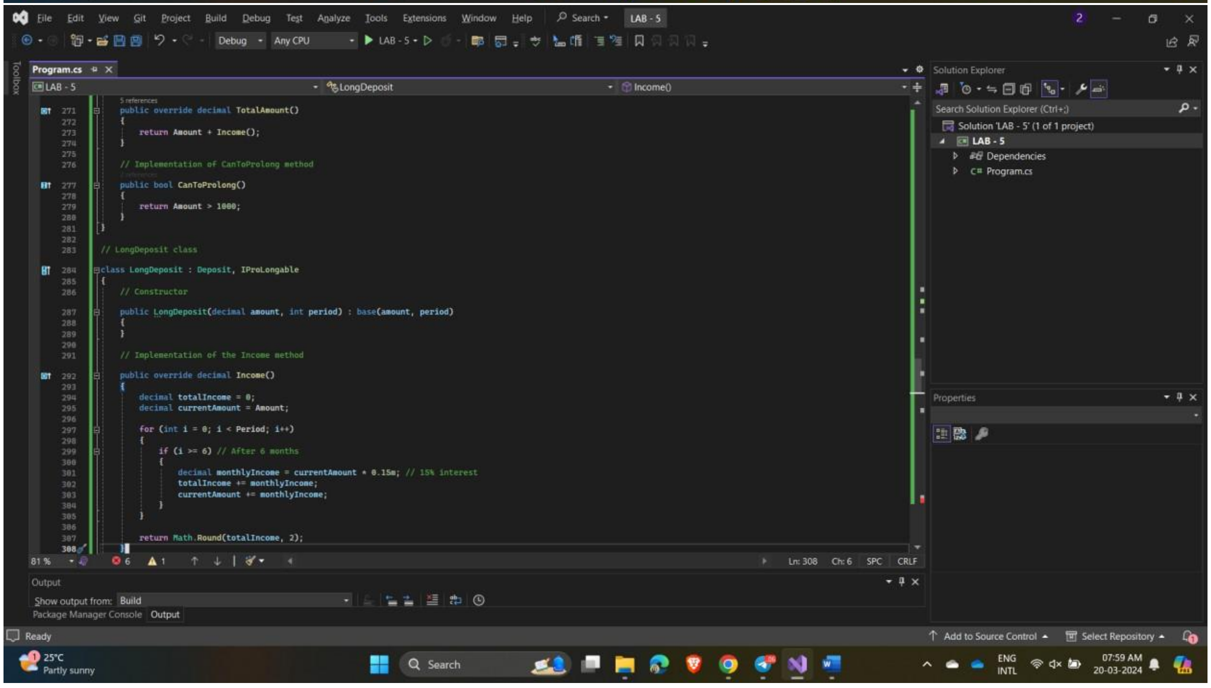
Entire Solution | 6 Errors | 1 Warning | 0 of 6 Messages | Build + IntelliSense

Code	Description	Project	File	Line	Suppression State
CS1026	) expected	LAB - 5	Program.cs	424	Active
CS1026	) expected	LAB - 5	Program.cs	424	Active
CS1002	; expected	LAB - 5	Program.cs	424	Active
CS1513	} expected	LAB - 5	Program.cs	424	Active
CS1513	} expected	LAB - 5	Program.cs	424	Active
	There is no argument given that corresponds to the required parameter 'amount' of				

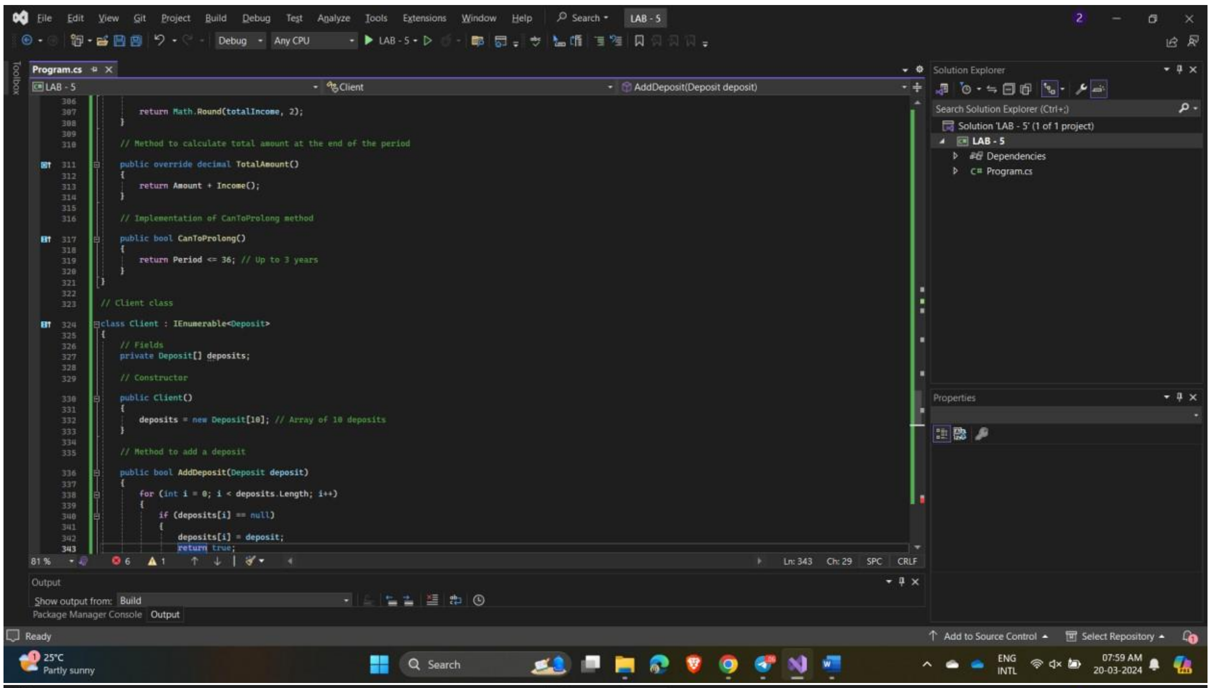
Ready



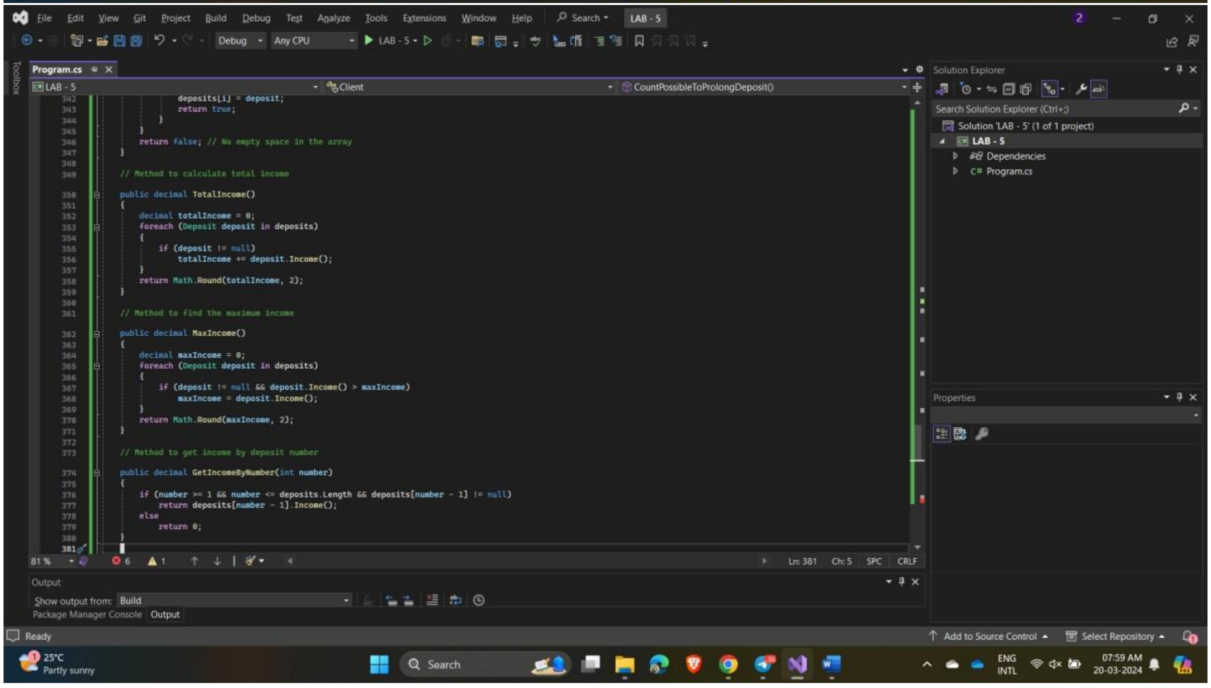
```
            return Math.Round(totalIncome, 2);
        }

        // Method to calculate total amount at the end of the period
        public override decimal TotalAmount()
        {
            return Amount + Income();
        }
    }

    // SpecialDeposit class
    class SpecialDeposit : Deposit, IProLongable
    {
        // Constructor
        public SpecialDeposit(decimal amount, int period) : base(amount, period)
        {
        }

        // Implementation of the Income method
        public override decimal Income()
        {
            decimal totalIncome = 0;
            decimal currentAmount = Amount;

            for (int i = 0; i < Period; i++)
            {
                decimal monthlyIncome = currentAmount * ((decimal)(i + 1) / 100); // Add percentage based on period
                totalIncome += monthlyIncome;
                currentAmount += monthlyIncome;
            }

            return Math.Round(totalIncome, 2);
        }

        // Method to calculate total amount at the end of the period
        public override decimal TotalAmount()
        {
```



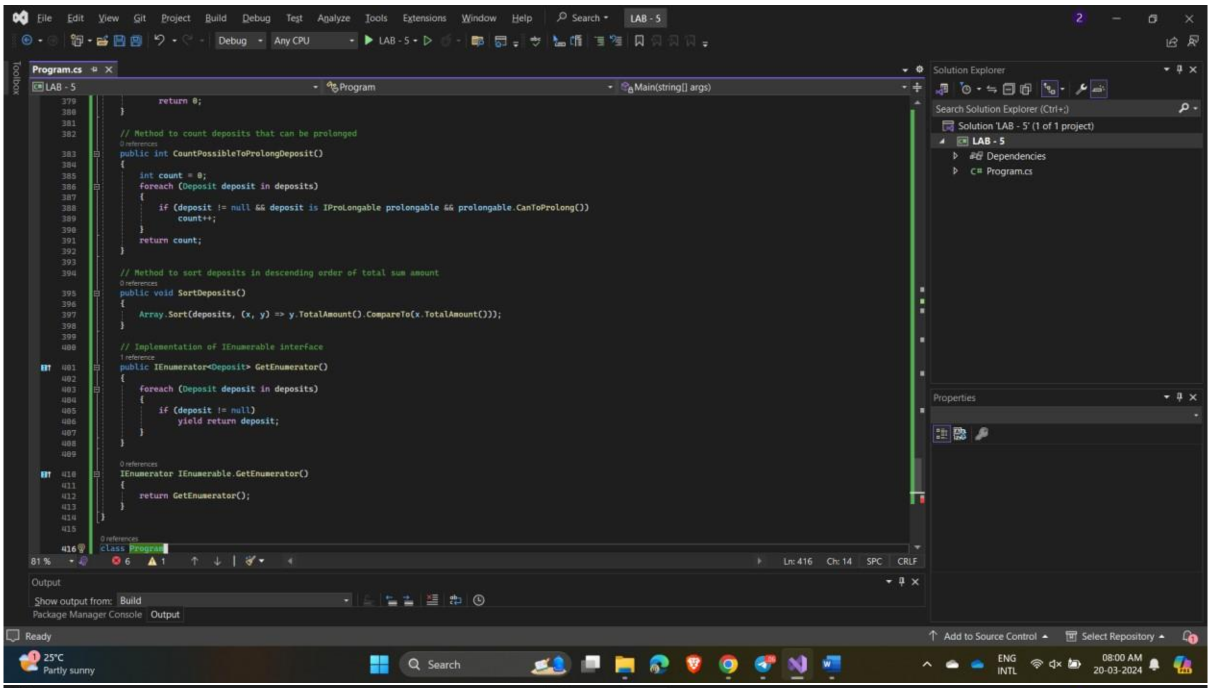
```
        public override decimal TotalAmount()
        {
            return Amount + Income();
        }

        // Implementation of CanToProlong method
        public bool CanToProlong()
        {
            return Amount > 1000;
        }
    }

    // LongDeposit class
    class LongDeposit : Deposit, IProLongable
    {
        // Constructor
        public LongDeposit(decimal amount, int period) : base(amount, period)
        {
        }

        // Implementation of the Income method
        public override decimal Income()
        {
            decimal totalIncome = 0;
            decimal currentAmount = Amount;

            for (int i = 0; i < Period; i++)
            {
                if (i >= 6) // After 6 months
                {
                    decimal monthlyIncome = currentAmount * 0.15m; // 15% interest
                    totalIncome += monthlyIncome;
                    currentAmount += monthlyIncome;
                }
            }

            return Math.Round(totalIncome, 2);
        }
```



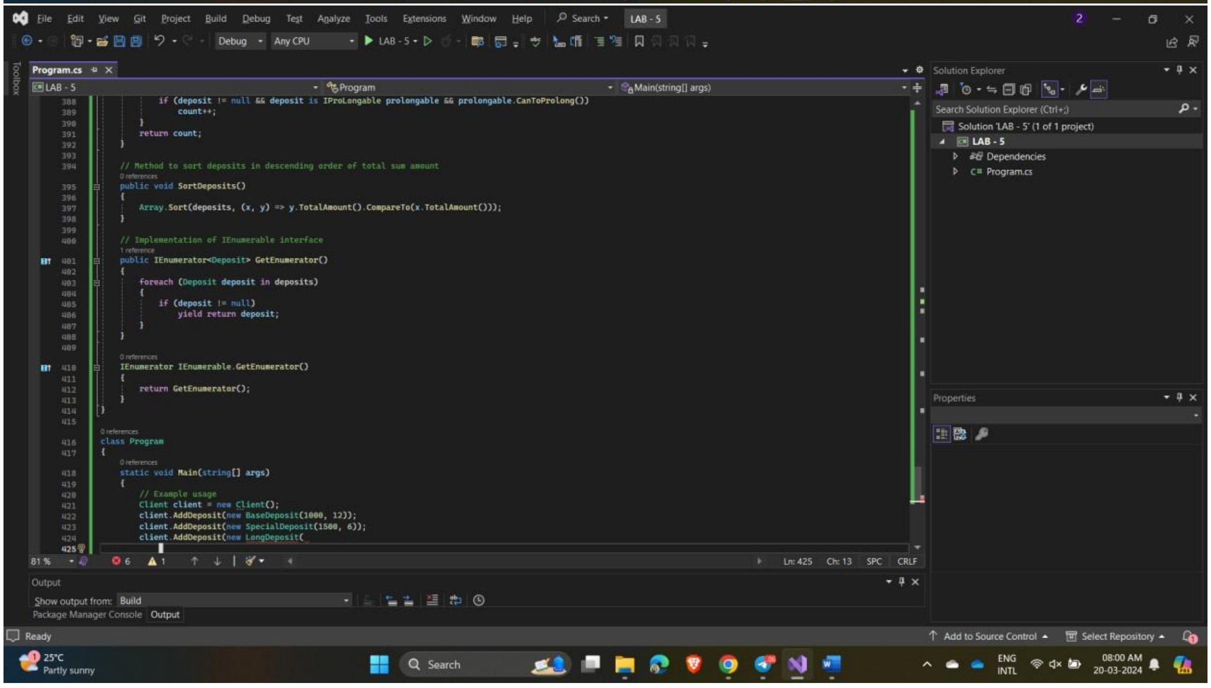
```
        return Math.Round(totalIncome, 2);
    }

    // Method to calculate total amount at the end of the period

    public override decimal TotalAmount()
    {
        return Amount + Income();
    }

    // Implementation of CanToProlong method

    public bool CanToProlong()
    {
        return Period <= 36; // Up to 3 years
    }
}

// Client class

class Client : IEnumerable<Deposit>
{
    // Fields
    private Deposit[] deposits;

    // Constructor

    public Client()
    {
        deposits = new Deposit[10]; // Array of 10 deposits
    }

    // Method to add a deposit

    public bool AddDeposit(Deposit deposit)
    {
        for (int i = 0; i < deposits.Length; i++)
        {
            if (deposits[i] == null)
            {
                deposits[i] = deposit;
                return true;
```



```
                deposits[i] = deposit;
                return true;
            }
        }
        return false; // No empty space in the array
    }

    // Method to calculate total income

    public decimal TotalIncome()
    {
        decimal totalIncome = 0;
        foreach (Deposit deposit in deposits)
        {
            if (deposit != null)
                totalIncome += deposit.Income();
        }
        return Math.Round(totalIncome, 2);
    }

    // Method to find the maximum income

    public decimal MaxIncome()
    {
        decimal maxIncome = 0;
        foreach (Deposit deposit in deposits)
        {
            if (deposit != null && deposit.Income() > maxIncome)
                maxIncome = deposit.Income();
        }
        return Math.Round(maxIncome, 2);
    }

    // Method to get income by deposit number

    public decimal GetIncomeByNumber(int number)
    {
        if (number >= 1 && number <= deposits.Length && deposits[number - 1] != null)
            return deposits[number - 1].Income();
        else
            return 0;
    }

```

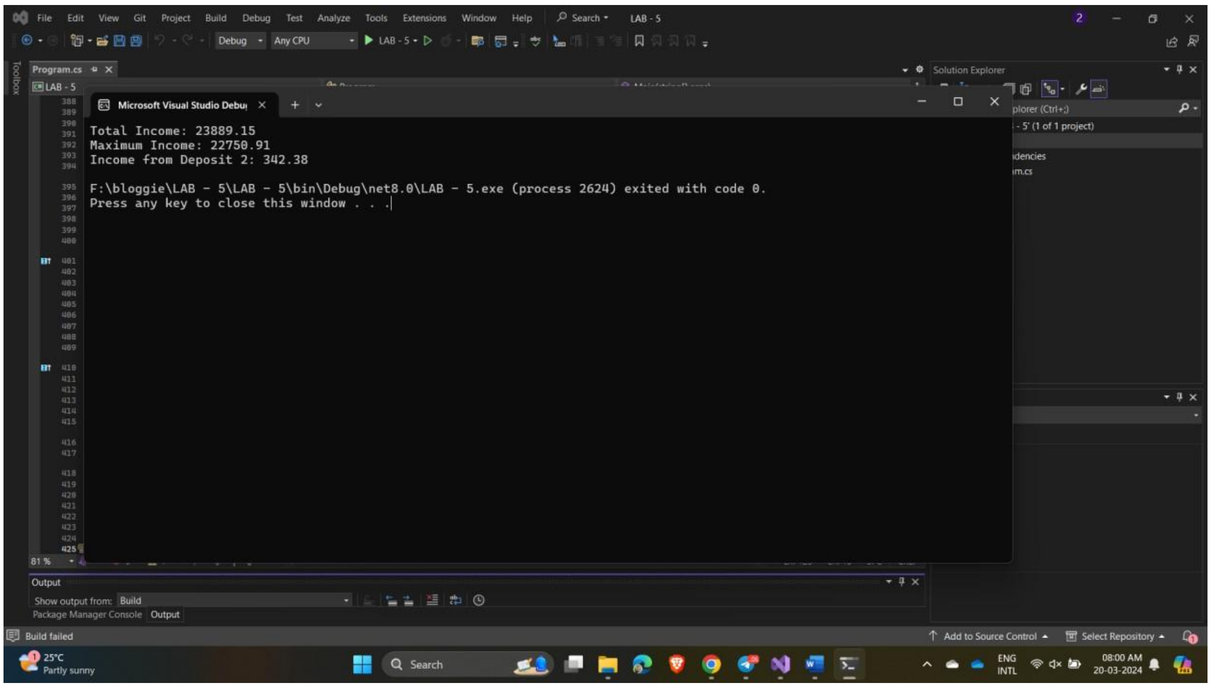


```
            return 0;
        }

    // Method to count deposits that can be prolonged
    public int CountPossibleToProlongDeposit()
    {
        int count = 0;
        foreach (Deposit deposit in deposits)
        {
            if (deposit != null && deposit is IProLongable prolongable && prolongable.CanToProlong())
                count++;
        }
        return count;
    }

    // Method to sort deposits in descending order of total sum amount
    public void SortDeposits()
    {
        Array.Sort(deposits, (x, y) => y.TotalAmount().CompareTo(x.TotalAmount()));
    }

    // Implementation of IEnumerable interface
    public IEnumerator<Deposit> GetEnumerator()
    {
        foreach (Deposit deposit in deposits)
        {
            if (deposit != null)
                yield return deposit;
        }
    }

    IEnumerator IEnumerable.GetEnumerator()
    {
        return GetEnumerator();
    }
}

class Program
```



```
            if (deposit != null && deposit is IProLongable prolongable && prolongable.CanToProlong())
                count++;
        }
        return count;
    }

    // Method to sort deposits in descending order of total sum amount
    public void SortDeposits()
    {
        Array.Sort(deposits, (x, y) => y.TotalAmount().CompareTo(x.TotalAmount()));
    }

    // Implementation of IEnumerable interface
    public IEnumerator<Deposit> GetEnumerator()
    {
        foreach (Deposit deposit in deposits)
        {
            if (deposit != null)
                yield return deposit;
        }
    }

    IEnumerator IEnumerable.GetEnumerator()
    {
        return GetEnumerator();
    }
}

class Program
{
    static void Main(string[] args)
    {
        // Example usage
        Client client = new Client();
        client.AddDeposit(new BaseDeposit(1000, 12));
        client.AddDeposit(new SpecialDeposit(1500, 6));
        client.AddDeposit(new LongDeposit(
```

OUTPUT :


```
Total Income: 23889.15
Maximum Income: 22750.91
Income from Deposit 2: 342.38

F:\bloggie\LAB - 5\LAB - 5\bin\Debug\net8.0\LAB - 5.exe (process 2624) exited with code 0.
Press any key to close this window . . .
```