Text Editor

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

public class Program
{
    // Stack to store undo history
    public static Stack<string> undoStack = new Stack<string>();

    public static void Main(string[] args)
    {
        TextEditor editor = new TextEditor();
        Console.WriteLine("Enter the text");
        string text = Console.ReadLine();

        Console.WriteLine("Enter yes, if you need to delete the last letter in your sentence");
        string deleteResponse = Console.ReadLine();

        if (deleteResponse.ToLower() == "yes")
        {
            text = editor.DeleteLastCharacter(text);
            Console.WriteLine("Your sentence after deletion of last letter is :");
            Console.WriteLine(text);
        }
        else
        {
            Console.WriteLine("Your editing completed");
            return;
        }
```

```
        Console.WriteLine("Enter yes, if you need to undo the last deleted letter");
        string undoResponse = Console.ReadLine();

        if (undoResponse.ToLower() == "yes")
        {
            text = editor.UndoDeletion();
            Console.WriteLine("Your sentence after undoing the deletion is :");
            Console.WriteLine(text);
        }
        else
        {
            Console.WriteLine("Your editing completed");
        }
    }
}

public class TextEditor
{
    public string DeleteLastCharacter(string text)
    {
        if (text.Length > 0)
        {
            // Add current text to undo stack
            Program.undoStack.Push(text);

            // Delete the last character
            text = text.Substring(0, text.Length - 1);
        }
        return text + "_";
    }
```

```
    public string UndoDeletion()
    {
        if (Program.undoStack.Count > 0)
        {
            // Retrieve the previous text
            string previousText = Program.undoStack.Pop();
            return previousText;
        }
        return "_";
    }
}
```

Movie studio

```
using System;

public class Program
{
    public static void Main(string[] args)
    {
        Console.WriteLine("Enter Movie Genre (Action/Drama/Comedy):");
        string genre = Console.ReadLine();

        Console.WriteLine("Enter Main Actor's Salary:");
        double mainActorSalary;
        while(!double.TryParse(Console.ReadLine(), out mainActorSalary))
        {
            Console.WriteLine("Please enter a valid number for the main actor's salary.");
        }
```

```
        MovieUtility movieUtility = new MovieUtility();

        if (movieUtility.ValidateMovieSpecification(genre, mainActorSalary))
        {
            double productionCost = movieUtility.CalculateProductionCost(genre, mainActorSalary);
            Console.WriteLine($"Production Cost: {productionCost}");
        }
        else
        {
            Console.WriteLine("Invalid movie specifications");
        }
    }
}

public class MovieUtility
{
    public bool ValidateMovieSpecification(string genre, double mainActorSalary)
    {
        if ((genre == "Action" || genre == "Drama" || genre == "Comedy") && mainActorSalary > 0)
        {
            return true;
        }
        return false;
    }

    public double CalculateProductionCost(string genre, double mainActorSalary)
    {
        double productionCost = 0;

        switch (genre)
```

```
        {
            case "Action":
                productionCost = 500000 + (mainActorSalary * 3);
                break;
            case "Drama":
                productionCost = 300000;
                break;
            case "Comedy":
                productionCost = 200000;
                break;
        }

        return productionCost;
    }
}
```

Sales performance

```
using System;

public class Program
{
    public static void Main(string[] args)
    {
        Console.WriteLine("Enter the number of days");
        int numberOfDays;
        while (!int.TryParse(Console.ReadLine(), out numberOfDays) || numberOfDays <= 0)
        {
            Console.WriteLine("Please enter a valid number of days greater than 0.");
        }
```

```
    double[] salesAmounts = new double[numberOfDays];
    for (int i = 0; i < numberOfDays; i++)
    {
        Console.WriteLine($"Enter the sales amount for Day {i + 1}");
        while (!double.TryParse(Console.ReadLine(), out salesAmounts[i]))
        {
            Console.WriteLine("Please enter a valid sales amount.");
        }
    }

    Console.WriteLine("Enter the sales target");
    double salesTarget;
    while (!double.TryParse(Console.ReadLine(), out salesTarget) || salesTarget <= 0)
    {
        Console.WriteLine("Please enter a valid sales target greater than 0.");
    }

    FindFirstQualifyingDay(salesAmounts, salesTarget);
  }

  public static void FindFirstQualifyingDay(double[] salesAmounts, double salesTarget)
  {
    for (int i = 0; i < salesAmounts.Length; i++)
    {
        if (salesAmounts[i] >= salesTarget)
        {
            Console.WriteLine($"Congratulations! The employee met or exceeded the sales target of
${salesTarget} on the {i + 1}-day");
            return;
        }
```

```
        }

        Console.WriteLine("No qualifying day found");
    }
}
```