EX 1

```
using System;

class JaggedRowSums

{

   static void Main()

   {

      int[][] jagged =

      {

         new[] { 1, 2, 3 },

         new[] { 4, 5 },

         new int[] { },

         null,

         new[] { 10 }

      };

      for (int i = 0; i < jagged.Length; i++)

      {

         int sum = 0;

         string rowStr;

         if (jagged[i] == null)

         {

            rowStr = "null";

         }

         else

         {

            foreach (int num in jagged[i])

               sum += num;


            rowStr = "[" + string.Join(", ", jagged[i]) + "]";

         }

         Console.WriteLine($"Row {i}: {rowStr} => Sum = {sum}");

      } } }
```

EX 2a

```
using System;
class SwapWithRef
{
   static void Swap(ref int a, ref int b)
   {
      int temp = a;
      a = b;
      b = temp;
   }

   static void Main()
   {
      int x = 5;
      int y = 10;
      Console.WriteLine($"Before swap: x = {x}, y = {y}");
      Swap(ref x, ref y);
      Console.WriteLine($"After swap:  x = {x}, y = {y}");
   }
}
```

Ex 2b

```
using System;
class TriangleAreaInOut
{
   static void ComputeArea(in double b, in double h, out double area)
   {
      area = 0.5 * b * h;
   }
   static void Main()
   {
      Console.Write("Enter base: ");
      double b = Convert.ToDouble(Console.ReadLine());
      Console.Write("Enter height: ");
      double h = Convert.ToDouble(Console.ReadLine());
      ComputeArea(in b, in h, out double area);
      Console.WriteLine($"Base = {b}, Height = {h}, Area = {area}");
   }
}
```

Ex 3a

```
using System;
class Circle
{
   private double radius;
   public Circle(double r)
   {
      radius = r;
   }
   public double Area()
   {
```

```
return Math.PI * radius * radius;
    }
    public double Perimeter()
    {
        return 2 * Math.PI * radius;
    }
    static void Main()
    {
        Console.Write("Enter radius: ");
        double r = Convert.ToDouble(Console.ReadLine());
        Circle c = new Circle(r);
        Console.WriteLine($"Radius = {r}");
        Console.WriteLine($"Area = {c.Area()}");
        Console.WriteLine($"Perimeter = {c.Perimeter()}");
    }
}
```

Ex 3b

```
using System;

class Student
{
    public int Roll;
    public string Name;
    public int Mark1, Mark2, Mark3;

    public Student(int roll, string name, int m1, int m2, int m3)
    {
        Roll = roll;
        Name = name;
        Mark1 = m1;
```

```
        Mark2 = m2;
        Mark3 = m3;
    }
    public double AverageOfBestTwo()
    {
        int min = Math.Min(Mark1, Math.Min(Mark2, Mark3));
        int sum = Mark1 + Mark2 + Mark3;
        return (sum - min) / 2.0;
    }
    static void Main()
    {
        Console.Write("Enter Roll Number: ");
        int roll = Convert.ToInt32(Console.ReadLine());
        Console.Write("Enter Name: ");
        string name = Console.ReadLine();
        Console.Write("Enter Mark 1: ");
        int m1 = Convert.ToInt32(Console.ReadLine());
        Console.Write("Enter Mark 2: ");
        int m2 = Convert.ToInt32(Console.ReadLine());
        Console.Write("Enter Mark 3: ");
        int m3 = Convert.ToInt32(Console.ReadLine());
        Student s = new Student(roll, name, m1, m2, m3);
        Console.WriteLine($"\nRoll Number: {s.Roll}");
        Console.WriteLine($"Name: {s.Name}");
        Console.WriteLine($"Average of Best Two Marks: {s.AverageOfBestTwo():F2}");
    }
}
```

Ex 4

```
using System;
class Time
{
   private int hours;
   private int minutes;
   private int seconds;
   public void GetTime()
   {
      Console.Write("Enter hours (1-12): ");
      hours = Convert.ToInt32(Console.ReadLine());
      if (hours < 1 || hours > 12)
         throw new Exception("Invalid hour.");
      Console.Write("Enter minutes (0-59): ");
      minutes = Convert.ToInt32(Console.ReadLine());
      if (minutes < 0 || minutes > 59)
         throw new Exception("Invalid minute.");
      Console.Write("Enter seconds (0-59): ");
      seconds = Convert.ToInt32(Console.ReadLine());
      if (seconds < 0 || seconds > 59)
         throw new Exception("Invalid second.");
   }
   public void DisplayTime()
   {
      Console.WriteLine($"Time: {hours:D2}:{minutes:D2}:{seconds:D2}");
   }
}

class Program
{
   static void Main()
```

```
    {
        try
        {
            Time t = new Time();
            t.GetTime();
            t.DisplayTime();
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
}
```

Ex 5

```
using System;
delegate void TrafficDel();
class TrafficSignal
{
    public static void Yellow()
    {
        Console.WriteLine("Yellow Light Signal To Get Ready");
    }
    public static void Green()
    {
        Console.WriteLine("Green Light Signal To Go");
    }
    public static void Red()
    {
        Console.WriteLine("Red Light Signal To Stop");
```

```
    }
}

class Program
{
    static void Main()
    {
        TrafficDel signal;
        signal = TrafficSignal.Yellow;
        signal();
        signal = TrafficSignal.Green;
        signal();
        signal = TrafficSignal.Red;
        signal();
    }
}
```

Ex 6 -Department.cs

```
Public class Department
{  public int DeptId { get; set; }
   Public string DeptName { get; set; }
}
```

Program.cs

```
Using System;
Using System.Collections.Generic;
Using System.Linq;
Class Program
{
   Static List<Department> departments = new List<Department>();
```

```
Static void Main()
{
   Int choice;
   Do
   {
      Console.WriteLine("\n=== Department Management System ===");
      Console.WriteLine("1. Insert Department");
      Console.WriteLine("2. Display All Departments");
      Console.WriteLine("3. Update Department");
      Console.WriteLine("4. Delete Department");
      Console.WriteLine("5. Exit");
      Console.Write("Enter your choice: ")
      If (!int.TryParse(Console.ReadLine(), out choice))
      {
         Console.WriteLine("Invalid input! Please enter a number.");
         Continue;
      }
      Switch (choice)
      {
         Case 1:
            InsertDepartment();
            Break;
         Case 2:
            DisplayDepartments();
            Break;
         Case 3:
            UpdateDepartment();
            Break;
         Case 4:
            DeleteDepartment();
            Break;
```

```
            Case 5:
                Console.WriteLine("Exiting...");
                Break;
            Default:
                Console.WriteLine("Invalid choice. Try again.");
                Break;
        }
    } while (choice != 5);
  }
  Static void InsertDepartment()
  {
    Console.Write("Enter Department ID: ");
    Int id = Convert.ToInt32(Console.ReadLine());
    Console.Write("Enter Department Name: ");
    String name = Console.ReadLine();
    Departments.Add(new Department { DeptId = id, DeptName = name });
    Console.WriteLine("Department inserted successfully!");
}
  Static void DisplayDepartments()
  {
    If (departments.Count == 0)
    {
      Console.WriteLine("No departments available.");
      Return;
    }
    Console.WriteLine("\n--- Department List ---");
    Foreach (var dept in departments)
    {
      Console.WriteLine($"ID: {dept.DeptId}, Name: {dept.DeptName}");
    }
  }
```

```
  Static void UpdateDepartment()
  {
     Console.Write(“Enter Department ID to update: “);
     Int id = Convert.ToInt32(Console.ReadLine());
     Var dept = departments.FirstOrDefault(d => d.DeptId == id);
     If (dept == null)
     {
        Console.WriteLine(“Department not found!”);
        Return;
     }
     Console.Write(“Enter new Department Name: “);
     Dept.DeptName = Console.ReadLine();
     Console.WriteLine(“Department updated successfully!”);
  }
  Static void DeleteDepartment()
  {
     Console.Write(“Enter Department ID to delete: “);
     Int id = Convert.ToInt32(Console.ReadLine());
     Var dept = departments.FirstOrDefault(d => d.DeptId == id);
     If (dept == null)
     {
        Console.WriteLine(“Department not found!”);
        Return;
     }
     Departments.Remove(dept);
     Console.WriteLine(“Department deleted successfully!”);
  }
}
```