DAY13 ASSIGNMENT
BY
PAVAN KUMAR (09-02-2022)

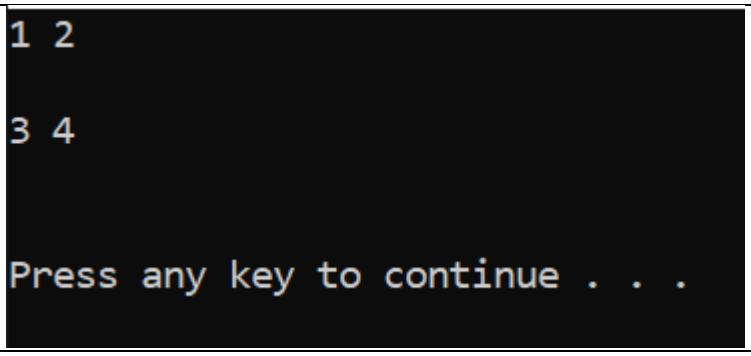
PROJECT: 1

Declare a 2-dimensional array of size (2,2) and initialize using indexes and print the values using nested for loop.

CODE:

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

namespace Day13Project1
{
   /// <summary>
   /// DONE BY: PAVAN
   /// PURPOSE: DECLARING A 2-D ARRAY WITH (2,2) SIZE MATRICES USING NESTED LOOP
   /// </summary>
   internal class Program
   {
      static void Main(string[] args)
      {
         int[,] data = new int[2,2];
         data[0,0] = 1;
         data[0,1] = 2;
         data[1,0] = 3;
         data[1,1] = 4;
         for(int i= 0; i<2; i++)

         {
            for (int j = 0; j < 2; j++)
               Console.Write(data[i,j]+ " ");
            Console.WriteLine("\n");
         }
         Console.ReadLine();
      }
   }
}
```

OUTPUT:



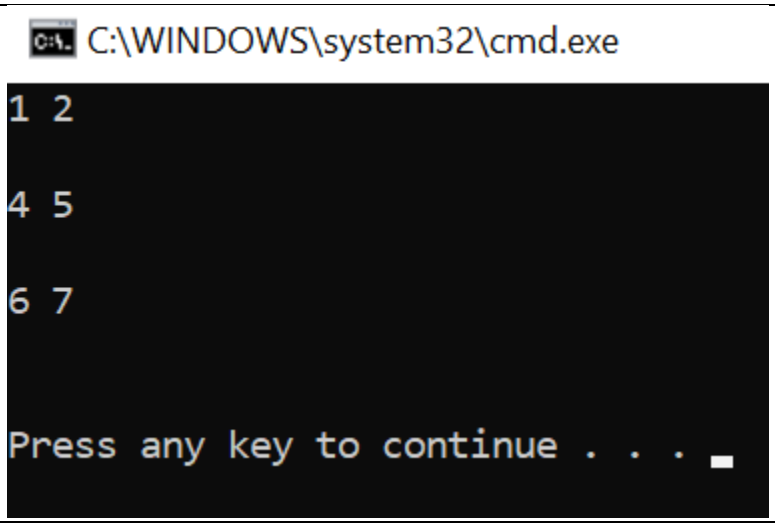
PROJECT: 2

Declare a 2-D array of size (3,2) and initialize in the same line while declaring and printing the values using nested for loop.

CODE:

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

namespace Day13Project1
{
   /// <summary>
   /// DONE BY: PAVAN
   /// PURPOSE: DECLARING A 2-D ARRAY WITH (3,2) SIZE MATRICES USING NESTED LOOP
   /// </summary>
   internal class Program
   {
      static void Main(string[] args)
      {
         int[,] data = new int[3, 2] {{1, 2}, {4, 5}, {6, 7}};

         for (int i = 0; i < 3; i++)

         {
            for (int j = 0; j < 2; j++)

               Console.Write(data[i, j] + " ");
            Console.WriteLine("\n");
         }
         Console.ReadLine();
      }
   }
}
```

OUPUT:



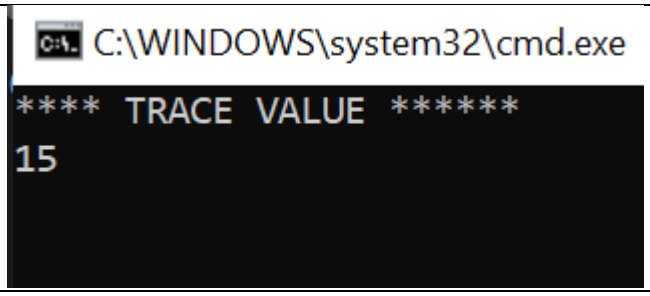
PROJECT:3

Declare a 2-D array of size (3,3) and print trace of the array.

CODE:

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

namespace Day13Project1
{
   /// <summary>
   /// DONE BY: PAVAN
   /// PURPOSE: DECLARING A 2-D ARRAY WITH (3,3) SIZE MATRICES USING NESTED LOOP FIND
TRACE VALUE:
   /// </summary>
   internal class Program
   {
      static void Main(string[] args)
      {
         int[,] data = new int[3, 3] {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
         int sum = 0;
         Console.WriteLine("**** TRACE VALUE ******");
         for (int i = 0; i < 3; i++)

         {
            for (int j = 0; j < 3; j++)
            {
               if(i== j)
                  sum = sum + data[i, j];
            }

         }
         Console.WriteLine(sum);
         Console.ReadLine();
      }
   }
}
```

OUTPUT:



PROJECT:4

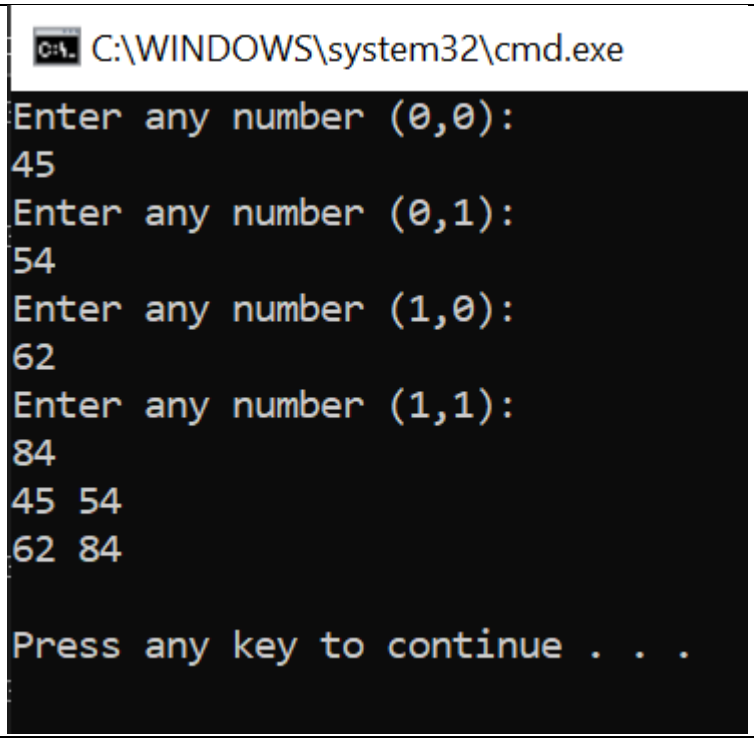
Declare a 2-D array of size (2,2) and read values from user and print the array values.

CODE:

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

```
namespace Day13Project1
{
   /// <summary>
   /// DONE BY: PAVAN
   /// PURPOSE: DECLARING A 2-D ARRAY WITH (2,2) SIZE MATRICES USING NESTED LOOP:
   /// </summary>
   internal class Program
   {
      static void Main(string[] args)
      {
         int[,] data = new int[2, 2];


         for (int i = 0; i < 2; i++)

         {
            for (int j = 0; j < 2; j++)
            {
               Console.WriteLine($"Enter any number ({i},{j}):  ");
               data[i,j] = Convert.ToInt32(Console.ReadLine());
            }

         }

         for (int i = 0; i < 2; i++)

         {
            for (int j = 0; j < 2; j++)
            {
               Console.Write(data[i,j] + " ");

            }
            Console.Write("\n");

         }
         Console.ReadLine();
      }
   }
}
```

OUTPUT:



PROJECT: 5

Declare TWO 2-D arrays of size (2,2) and read values from user and print the sum of the two matrices

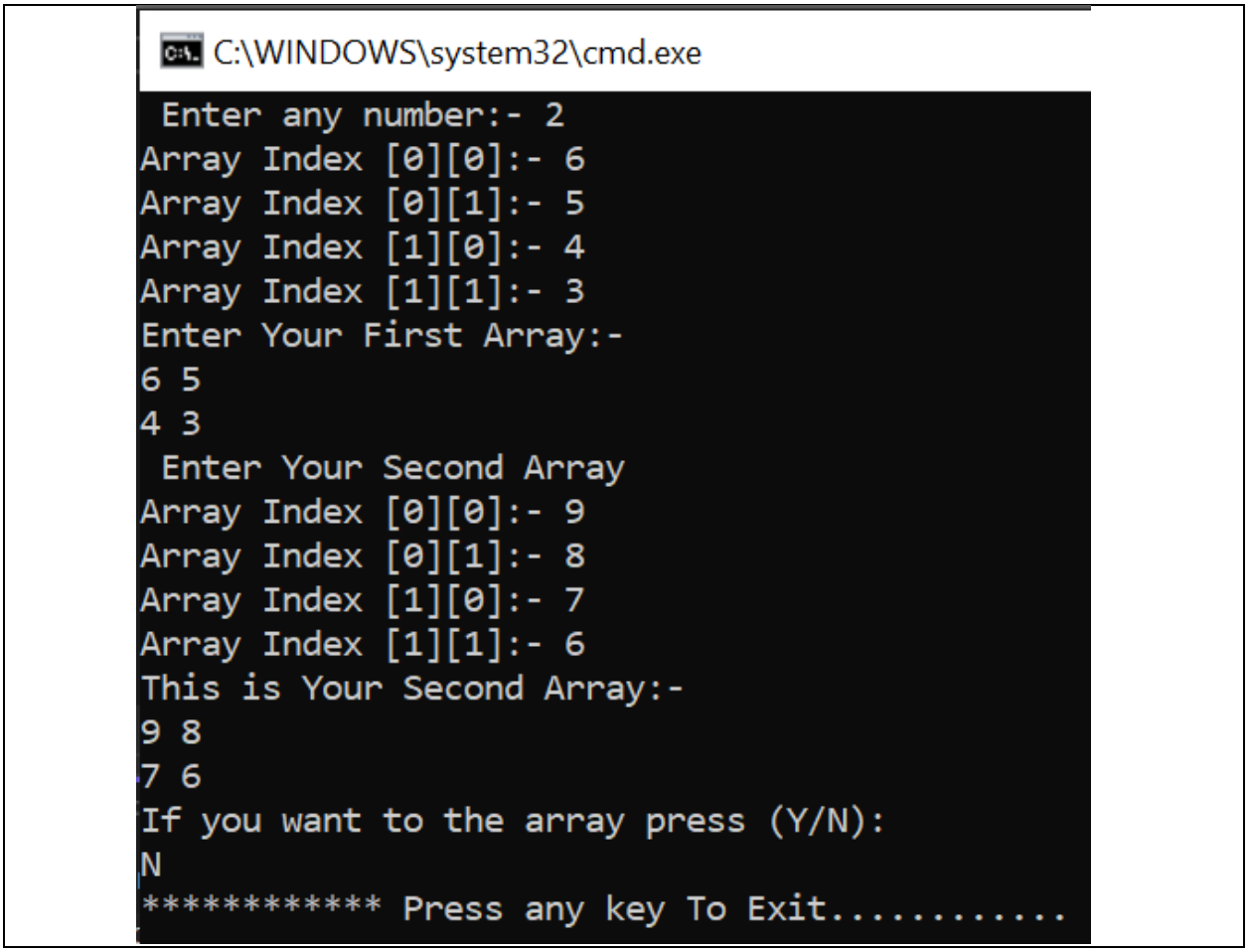
CODE:

```
using System;

 namespace Day13Project5
{
   /// <summary>
   /// DONE BY: PAVAN
   /// PURPOSE: Declare TWO 2-D arrays of size (2,2) and read values from user and print the sum of the two matrices
   /// </summary>
   internal class Program
   {
      static void Main(string[] args)
      {
         Program obj = new Program();
         obj.SumOfTwoArrays();
         Console.ReadLine();
      }
      void SumOfTwoArrays()
      {
         Console.Write(" Enter any number: - ");
         int arrayLength = Convert.ToInt32(Console.ReadLine());

         int[,] array = new int[arrayLength, arrayLength];
         int[,] arraySecond = new int[arrayLength, arrayLength];
         int[,] arraySum = new int[arrayLength, arrayLength];
```

```
for (int i = 0; i < arrayLength; i++)
{
   for (int j = 0; j < arrayLength; j++)
   {
      Console.Write("Array Index [{0}][{1}]: - ", i, j);
      array[i, j] = Convert.ToInt32(Console.ReadLine());
   }
}

Console.WriteLine("Enter Your First Array:-");

for (int i = 0; i < arrayLength; i++)
{
   for (int j = 0; j < arrayLength; j++)
   {
      if (j == 0)
      {
         Console.Write(array[i, j]);
      }
      else
      {
         Console.Write(" " + array[i, j]);
      }

   }
   Console.WriteLine();
}

Console.WriteLine(" Enter Your Second Array");

for (int i = 0; i < arrayLength; i++)
{
   for (int j = 0; j < arrayLength; j++)
   {
      Console.Write("Array Index [{0}] [{1}]: - ", i, j);
      arraySecond[i, j] = Convert.ToInt32(Console.ReadLine());
   }
}

Console.WriteLine("This is Your Second Array:-");

for (int i = 0; i < arrayLength; i++)
{
   for (int j = 0; j < arrayLength; j++)
   {
      if (j == 0)
      {
         Console.Write(arraySecond[i, j]);
      }
      else
      {
         Console.Write(" " + arraySecond[i, j]);
      }
```

```
            }
            Console.WriteLine();
        }

        Console.WriteLine("If you want to the array press (Y/N):");

        string userInput = Convert.ToString(Console.ReadLine());

        if (userInput.ToUpper() == "Y")
        {
            for (int i = 0; i < arrayLength; i++)
            {
                for (int j = 0; j < arrayLength; j++)
                {
                    arraySum[i, j] = array[i, j] + arraySecond[i, j];
                }
            }
            Console.WriteLine("Result is");
            for (int i = 0; i < arrayLength; i++)
            {
                for (int j = 0; j < arrayLength; j++)
                {
                    if (j == 0)
                    {
                        Console.Write(arraySum[i, j]);
                    }
                    else
                    {
                        Console.Write(" " + arraySum[i, j]);
                    }

                }
                Console.WriteLine();
            }
        }
        else
        {
            Console.WriteLine("********** Press any key To Exit.....");
        }
    }
  }
}
```

OUTPUT:



PROJECT: 6

Declare TWO 2-D arrays of size (2,2) and read values from user and print the product of the two matrices.

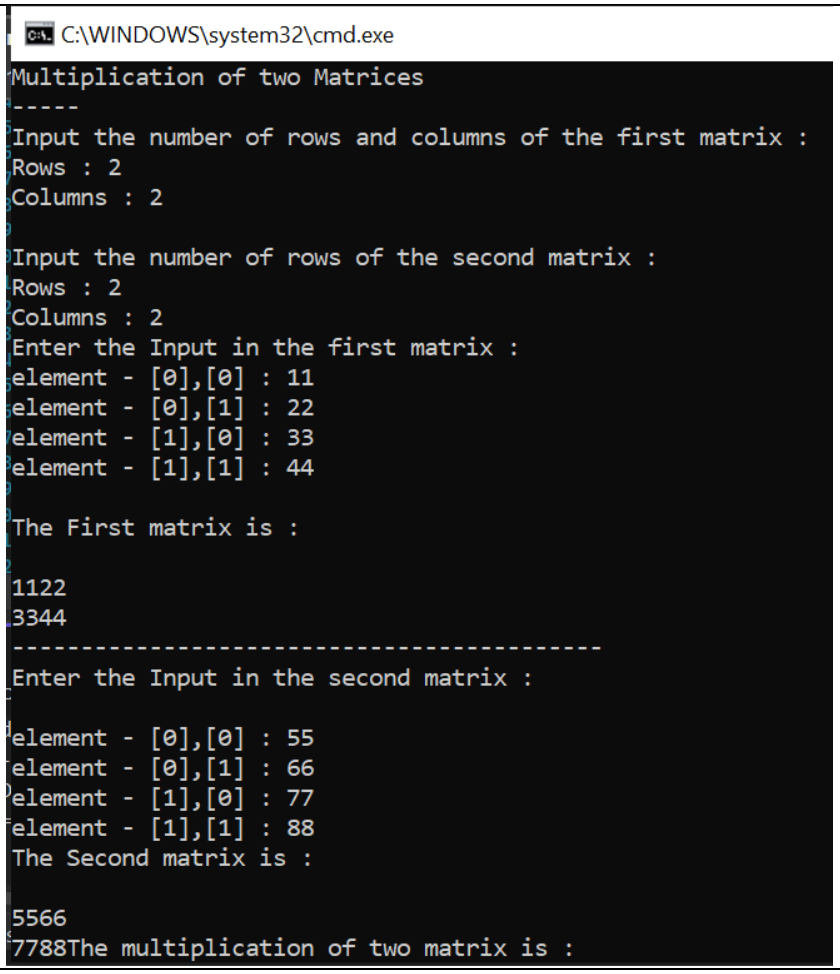
CODE:

```
using System;


namespace Day13Project6
{
   /// <summary>
   /// DONE BY: PAVAN
   /// PURPOSE: Declare TWO 2-D arrays of size (2,2) and read values from user and print the
product of the two matrices.
   /// </summary>
   internal class Program
   {
      static void Main(string[] args)
      {
         int i, j, k, r1, c1, r2, c2, sum = 0;

         int[,] arr1 = new int[50, 50];
         int[,] brr1 = new int[50, 50];
         int[,] crr1 = new int[50, 50];
```

```
Console.Write("Multiplication of two Matrices");
Console.Write("\n-----\n");
Console.Write("Input the number of rows and columns of the first matrix :\n");
Console.Write("Rows : ");
r1 = Convert.ToInt32(Console.ReadLine());
Console.Write("Columns : ");
c1 = Convert.ToInt32(Console.ReadLine());

Console.Write("\nInput the number of rows of the second matrix:\n");
Console.Write("Rows: ");
r2 = Convert.ToInt32(Console.ReadLine());
Console.Write("Columns: ");
c2 = Convert.ToInt32(Console.ReadLine());

if (c1! = r2)
{
   Console.Write("Multiplication of Matrix is not possible.");
   Console.Write("Column of first matrix and row of second matrix must be same.");
}
else
{
   Console.Write("Enter the Input in the first matrix :\n");
   for (i = 0; i < r1; i++)
   {
      for (j = 0; j < c1; j++)
      {
         Console.Write($"element - [{i}],[{j}]: ");
         arr1[i, j] = Convert.ToInt32(Console.ReadLine());
      }
   }

   Console.Write("\nThe First matrix is:\n");
   for (i = 0; i < r1; i++)
   {
      Console.Write("\n");
      for (j = 0; j < c1; j++)
         Console.Write("{0}", arr1[i, j]);
   }
   Console.Write("\n-----------------------------------------\n");
   Console.Write("Enter the Input in the second matrix:\n\n");
   for (i = 0; i < r2; i++)
   {
      for (j = 0; j < c2; j++)
      {
         Console.Write("element - [{0}],[{1}]: ", i, j);
         brr1[i, j] = Convert.ToInt32(Console.ReadLine());
      }
   }

   Console.Write("The Second matrix is:\n");
   for (i = 0; i < r2; i++)
   {
      Console.Write("\n");
```

```
            for (j = 0; j < c2; j++)
               Console.Write("{0}", brr1[i, j]);
         }
         ///summary
         ///Multiplication of Two Matrices///
         for (i = 0; i < r1; i++)
            for (j = 0; j < c2; j++)
               crr1[i, j] = 0;
         for (i = 0; i < r1; i++)
         {
            for (j = 0; j < c2; j++)
            {
               sum = 0;
               for (k = 0; k < c1; k++)
                  sum = sum + arr1[i, k] * brr1[k, j];
               crr1[i, j] = sum;
            }
         }
         Console.Write("The multiplication of two matrix is: \n");
         for (i = 0; i < r1; i++)
         {

            for (j = 0; j < c2; j++)
            {
               Console.Write("{0}", crr1[i, j]);
            }
         }
      }
   }
}
```

OUTPUT:



Q7). What is a jagged array? What is the benefit of jagged array

JAGGED ARRAY:

A jagged array is an array whose elements are arrays of different sizes. A jagged array is sometimes called an "**array of arrays**."

SYNTAX: **int**[][] arr = **new int**[2][];

BENEFITS OF JAGGED ARRAY:

- To save memory we use a jagged array.
- Jagged Array can also be mixed with multidimensional arrays.
- One of the most crucial advantages is that it makes things easy where there is a need to store data in a multidimensional way using the same variable name.

PROJECT:8

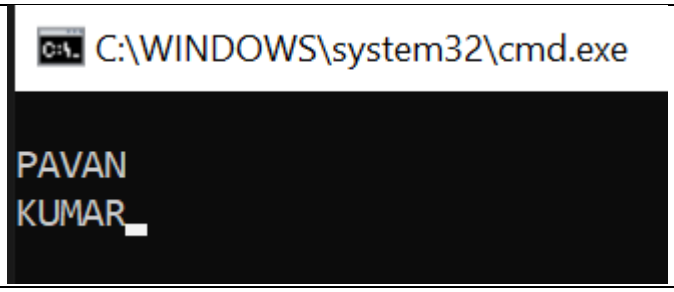
WACP to declare a jagged array and print values

CODE:

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

```
namespace Day13Project7
{
   /// <summary>
   /// DONE BY: PAVAN
   /// PURPOUSE: JAGGED ARRAY//
   /// </summary>
   internal class Program
   {
      static void Main(string[] args)
      {
         char[][] names = new char[2][];
         names[0] = new char[] {'P', 'A', 'V', 'A', 'N'};
         names[1] = new char[] {'K', 'U', 'M','A', 'R'};

         Console.WriteLine(" ");
         for (int i = 0; i < names.Length; i++)
         {
            for (int j = 0; j < names[i].Length; j++)
               Console.Write(names[i][j]);
            Console.ReadLine();
         }
      }
   }
}
```

OUTPUT:



Q9). What is Recursion

A function that calls another function is normal but when a function calls itself then that is a recursive function.

The recursion property depends on two things:

- ★ The satisfaction of the condition, which enables it to call itself again and again.
- ★ The calling of function or method with parameters and receiving new parameter after every execution.

PROJECT: 10

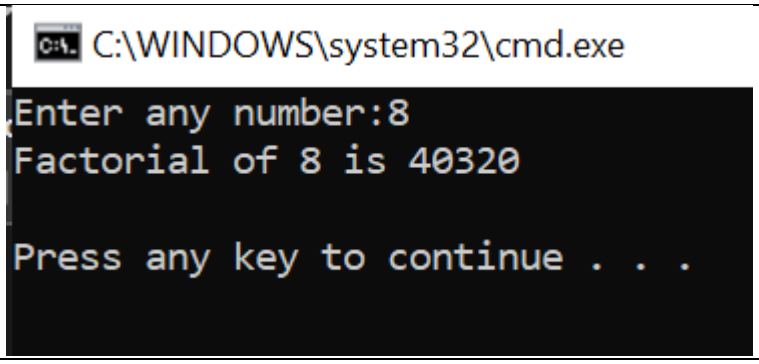
WACP to illustrate usage of Recursion. What are the benefits of recursion

CODE:

```
using System;
```

```
using System.Collections.Generic;

namespace RecursionFunction
{
   /// <summary>
   /// DONE BY: PAVAN
   /// PURPOUSE: USING RECURSION FIND FACTORIAL//
   /// </summary>
   internal class Program
   {
     public static void PrintOutput(int n)
      {
         Console.WriteLine("Factorial of {0} is {1}", n, Factorial(n));
      }
      public static int Factorial(int input)
      {
         if (input == 0)
            return 1;
         else
            return input * Factorial(input - 1);
      }
      static void Main(string[] args)
      {
         int input;
         Console.Write("Enter any number:");
         input = Convert.ToInt32(Console.ReadLine());

         PrintOutput(input);
         Console.ReadLine();
      }
   }
}
```

OUTPUT:



PROJECT: 11

WACP to illustrate usage of Stack<>. Write a couple of points about Stack

STACK:

Stack is a special type of collection that stores elements in LIFO style (Last in First Out).

- It comes under System.Collection.Generic namespace.
- Elements can be added using the Push() method.

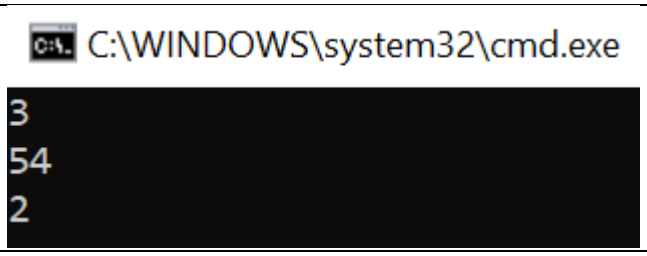
- Elements can be retrieved using the Pop() and the Peek() methods.

Stack<int> data = new Stack<int>();

CODE:

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

namespace Day13Project1
{
   /// <summary>
   /// DONE BY: PAVAN
   /// PURPOSE: USING STACK
   /// </summary>
   internal class Program
   {
      static void Main(string[] args)
      {
         Stack<int> data = new Stack<int>();
         data.Push(16);
         data.Push(78);
         data.Push(54);
         Console.WriteLine(data.Count);
         Console.WriteLine(data.Pop());
         Console.WriteLine(data.Count);
         Console.ReadLine();
      }
   }
}
```

OUTPUT:



PROJECT:12

WACP to illustrate usage of Queue<>. Write a couple of points about Queue.

QUEUE:

Queue is a special type of collection that stores the elements in FIFO style (First in First Out).

- It comes under System.Collection.Generic namespace.
- Elements can be added using the Enqueue() method.
- Elements can be retrieved using the Dequeue() and the Peek() methods.

Queue<int> data= new Queue<int>();

CODE:

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

```
namespace Day13Project1
{
   /// <summary>
   /// DONE BY: PAVAN
   /// PURPOSE: USING QUEUE
   /// </summary>
   internal class Program
   {
      static void Main(string[] args)
      {
         Queue<int> data = new Queue<int>();
         data.Enqueue(16);
         data.Enqueue(78);
         data.Enqueue(54);
         Console.WriteLine(data.Count);
         Console.WriteLine(data.Dequeue());
         Console.WriteLine(data.Count);
         Console.ReadLine();
      }
   }
}
```

OUTPUT: