

1. (a) Write a method in C# that takes an array as an input parameter and uses two methods, one to find the largest array element and other to compute the average of array elements.

Aim :

To write a method in C# that takes an array as an input parameter and uses two methods, one to find the largest array element and other to compute the average of array elements.

Algorithm :

1. Start Program
2. Declare the necessary variables.
3. This program takes n number of elements from the user and stores it in the arr array. To find the largest element,
4. the first two elements of array are checked and the largest of these two elements are placed in arr[0]
5. The first and third elements are checked and largest of these two elements is placed in arr[0]. This process continues until the first and last elements are checked
6. The largest number will be stored in the arr[0] position
7. Declare new variable sum as 0 and add elements in the array and store in sum
8. divide sum by total number of elements in array
9. return the results and print the result.

Program :

**using System;**

**namespace CS1A**

**{**

**class ArrayLargest**

**{**

**int large(int[] a, int n)**

**{**

**int large = a[0];**

**for (int i = 1; i < n; i++)**

**{**

**if (a[i] > large)**

**large = a[i];**

**}**

**return large;**

**}**

**float avg(int[] a, int n)**

**{**

**float avg = 0;**

**for (int i = 0; i < n; i++)**

**avg += a[i];**

**avg /= n;**

**return avg;**

**}**

**public static void Main()**

**{**

**ArrayLargest obj = new ArrayLargest();**

**int n;**

**float large, avg;**

**int[] a = new int[50];**

**Console.Write("Enter the size of Array : ");**

**string s, s1;**

**s = Console.ReadLine();**

**n = Int32.Parse(s);**

**Console.WriteLine("Enter the array elements");**

**for (int i = 0; i < n; i++)**

**{**

**s1 = Console.ReadLine();**

**a[i] = Int32.Parse(s1);**

**}**

**Console.Write("");**

**large = obj.large(a, n);**

**avg = obj.avg(a, n);**

**Console.WriteLine("Largest element in the given array is {0}", large);**

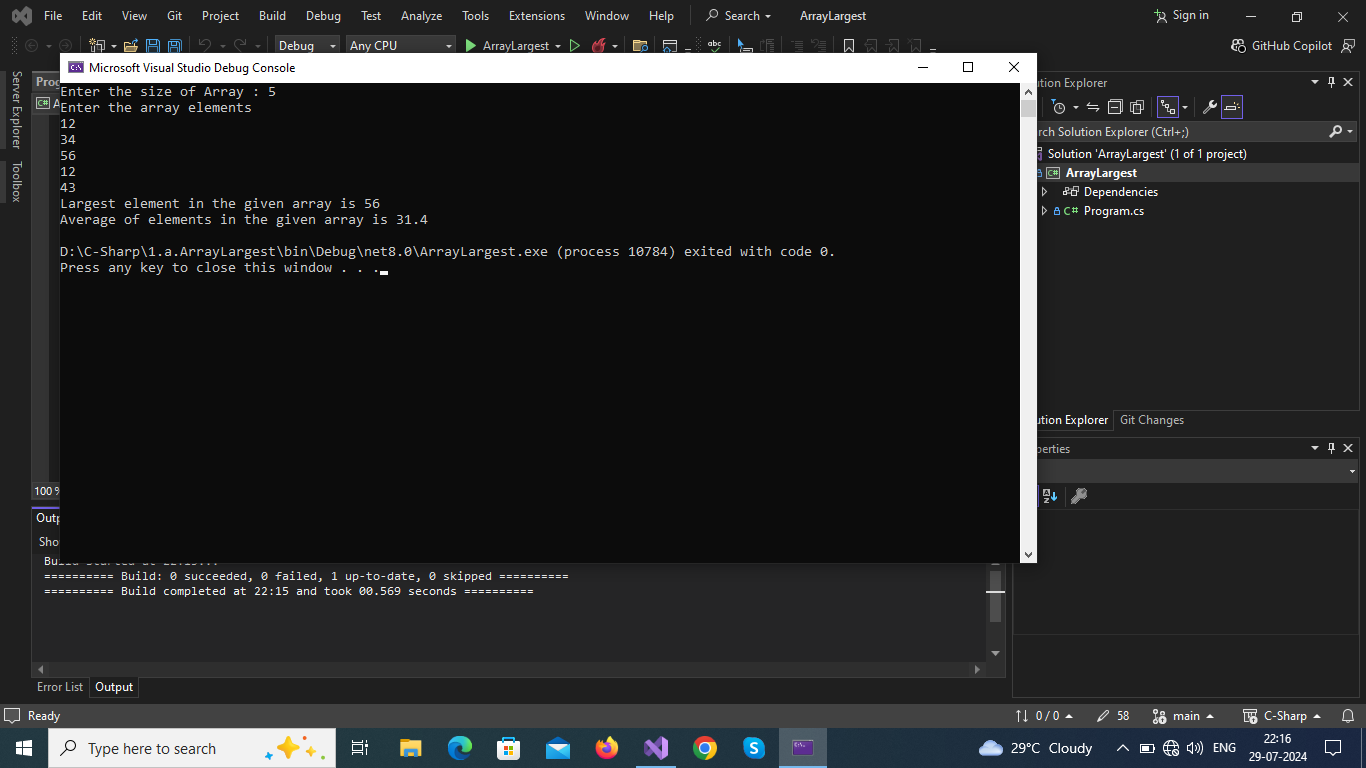
**Console.WriteLine("Average of elements in the given array is {0}", avg);**

**}**

**}**

**}**

**Output :**

****

**Result :**

Thus the program executed successfully and ouput was verified.

1. (b) Write a C# program to find the sum of all the elements present in a jagged array

Aim:

To write a C# program to find the sum of all the elements present in a jagged array

Algorithm:

1. Start Program
2. Jagged Array : Collection of array is known with different size is known as jagged array
3. Each element of an array is accessed by a subscript operator([]).
4. Similiarly in jagged array each array is a element of jagged array and each array elements accessed by 2D accessing(ja[array\_index][array\_element\_index])
5. Declaration of jagged array:

Syntax

data\_type[][] variable\_name = new data\_type[Size\_of\_jagged\_array][]

1. Traverse over each elements of each array in jagged array add each of them together and store them in variable (eg;sum+=jagged\_array[i][j])
2. Print value of sum
3. Stop program

Program:

using System;

namespace CS1B

{

public class SumOfElementsInJaggedArray

{

static void Main(string[] args)

{

int sum = 0;

int n, val, k;

Console.Write("Enter How Many Arrays in Jagged Array : ");

n = Convert.ToInt32(Console.ReadLine());

int[][] jagged\_array = new int[n][];

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

{

Console.Write("\nEnter How Many Elements in ({0}) - Array in Jagged Array : ", i + 1);

k = Convert.ToInt32(Console.ReadLine());

jagged\_array[i] = new int[k];

}

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

{

Console.WriteLine("\nEnter Elements of ({0}) - Array in Jagged Array ", i + 1);

for (int j = 0; j < jagged\_array[i].Length; j++)

{

Console.Write(" Enter Element {0} : ", j + 1);

jagged\_array[i][j] = Int32.Parse(Console.ReadLine());

}

}

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

{

val = 0;

for (int j = 0; j < jagged\_array[i].Length; j++)

{

val += jagged\_array[i][j];

}

Console.WriteLine("\nThe Sum of array ({0}) = {1}", i + 1, val);

sum += val;

}

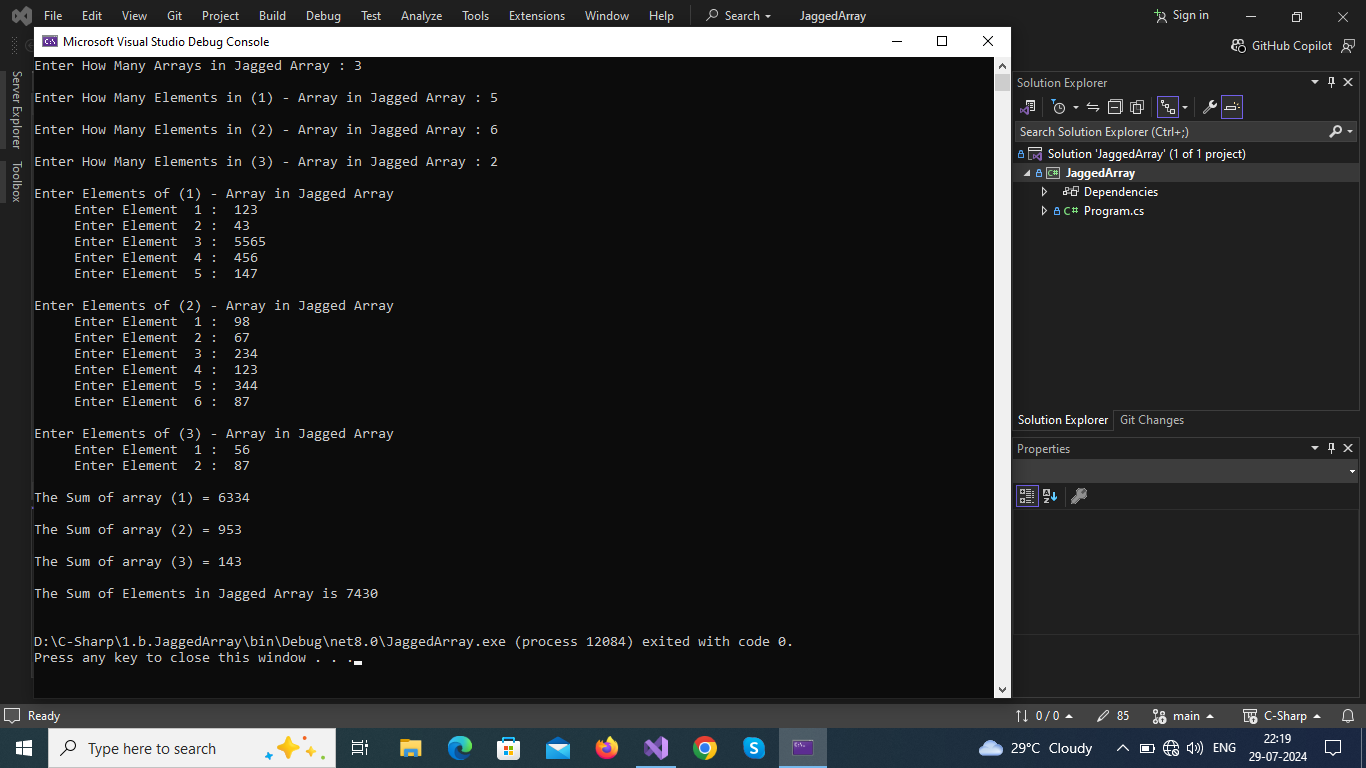
Console.WriteLine("\nThe Sum of Elements in Jagged Array is {0} \n", sum);

}

}

}

Output:



**Result :**

Thus the program executed successfully and ouput was verified.

2. (a) Write a program using while loop to reverse the digits of the number.

Aim:

To write a program using while loop to reverse the digits of the number

.

Algorithm :

1. Start Program
2. Get Input from user & store in variable
3. Declare a variable to store reversed number.
4. Using following logics we can perform reverse operation.
   1. To find last digit n%10;
   2. To find remaining part n/10;
5. To get reverse (use loop)
   1. reverse = reverve\*10 + lastdigit;
   2. number = number/10;
6. Print value of reverse
7. Stop program

Program :

**using System;**

**namespace CS2A**

**{**

**class NumberReverse**

**{**

**public static void Main(string[] args)**

**{**

**int n;**

**Console.Write("Enter an Integer : ");**

**n = Int32.Parse(Console.ReadLine());**

**int num = n;**

**int rev = 0;**

**while (n != 0)**

**{**

**rev = rev \* 10 + (n % 10);**

**n /= 10;**

**}**

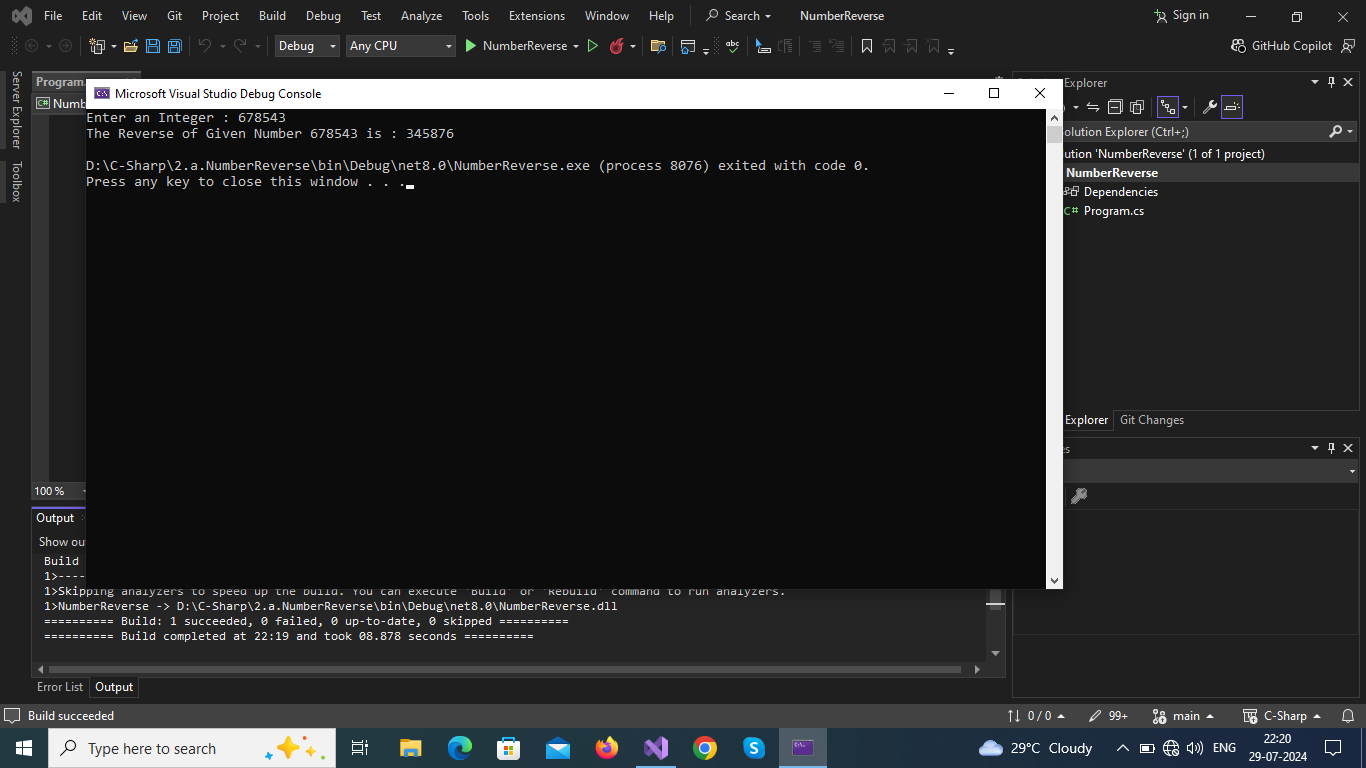
**Console.WriteLine("The Reverse of Given Number {0} is : {1} ", num, rev);**

**}**

**}**

**}**

**Output :**



**Result :**

Thus the program executed successfully and ouput was verified.

2. (b) Write a C# program that generates the Fibonacci series and prime numbers for the given number.

Aim:

To write a program that generates the Fibonacci series and prime numbers for the given number.

.

Algorithm :

1. Start Program
2. Write two methods :
   1. void fibo(param n) to generate Fibonacci series

logic => n1=0,n2=0

n3=n1+n2

n1=n2

n2=n3

* 1. void fibo(param n) to generate Prime Number

logic => use two loops to check whether the nmber is prime or not if prime print it.

1. Print value of results
2. Stop program

Program :

**using System;**

**namespace CS2B**

**{**

**class FiboPrime**

**{**

**void Prime(int n)**

**{**

**int c, k = 0;**

**for (int i = 1; i < n; k++)**

**{**

**c = 0;**

**int j = 1;**

**for (j = 1; j <= k; j++)**

**{**

**if (k % j == 0)**

**{**

**c++;**

**}**

**}**

**if (c == 2)**

**{**

**Console.WriteLine(k);**

**i++;**

**}**

**}**

**}**

**void Fibo(int n)**

**{**

**ulong n1 = 0, n2 = 1, n3;**

**Console.WriteLine(" {0}\n {1}", n1, n2);**

**for (int i = 2; i <= n; i++)**

**{**

**n3 = n1 + n2;**

**Console.WriteLine(" " + n3);**

**n1 = n2;**

**n2 = n3;**

**}**

**}**

**public static void Main(string[] args)**

**{**

**int n;**

**Console.Write("Enter Interger to generate Fibonacci Series or Prime Number : ");**

**n = Int32.Parse(Console.ReadLine());**

**Console.Write("1.Fibonacci Series\n2.Prime Numbers\nChoose Option : ");**

**int ch = Int32.Parse(Console.ReadLine());**

**FiboPrime fb = new FiboPrime();**

**switch (ch)**

**{**

**case 1:**

**Console.WriteLine("\n{0} Fibonacci Terms ", n);**

**fb.Fibo(n);**

**break;**

**case 2:**

**Console.WriteLine("\n{0} Prime Numbers ", n);**

**fb.Prime(n);**

**break;**

**default:**

**Console.WriteLine("Wrong Choice ");**

**Main(args);**

**break;**

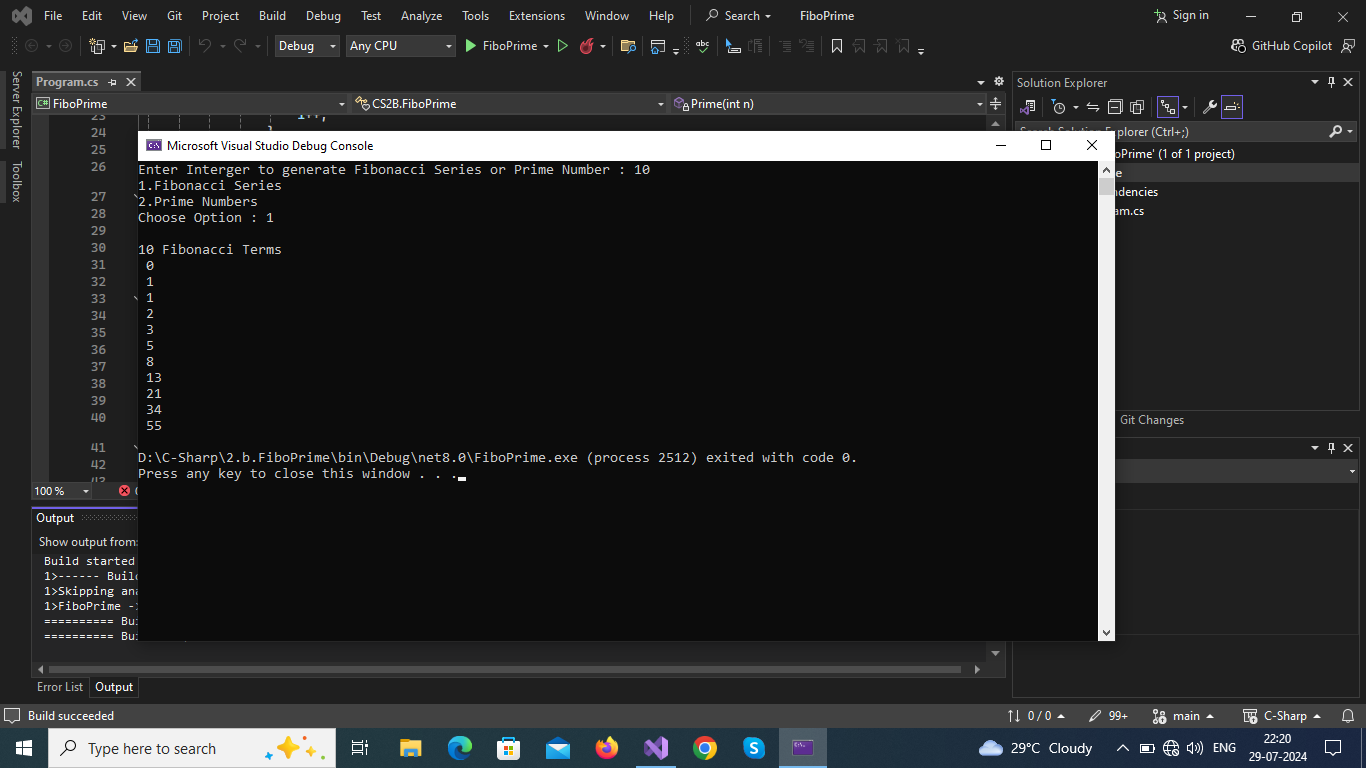
**}**

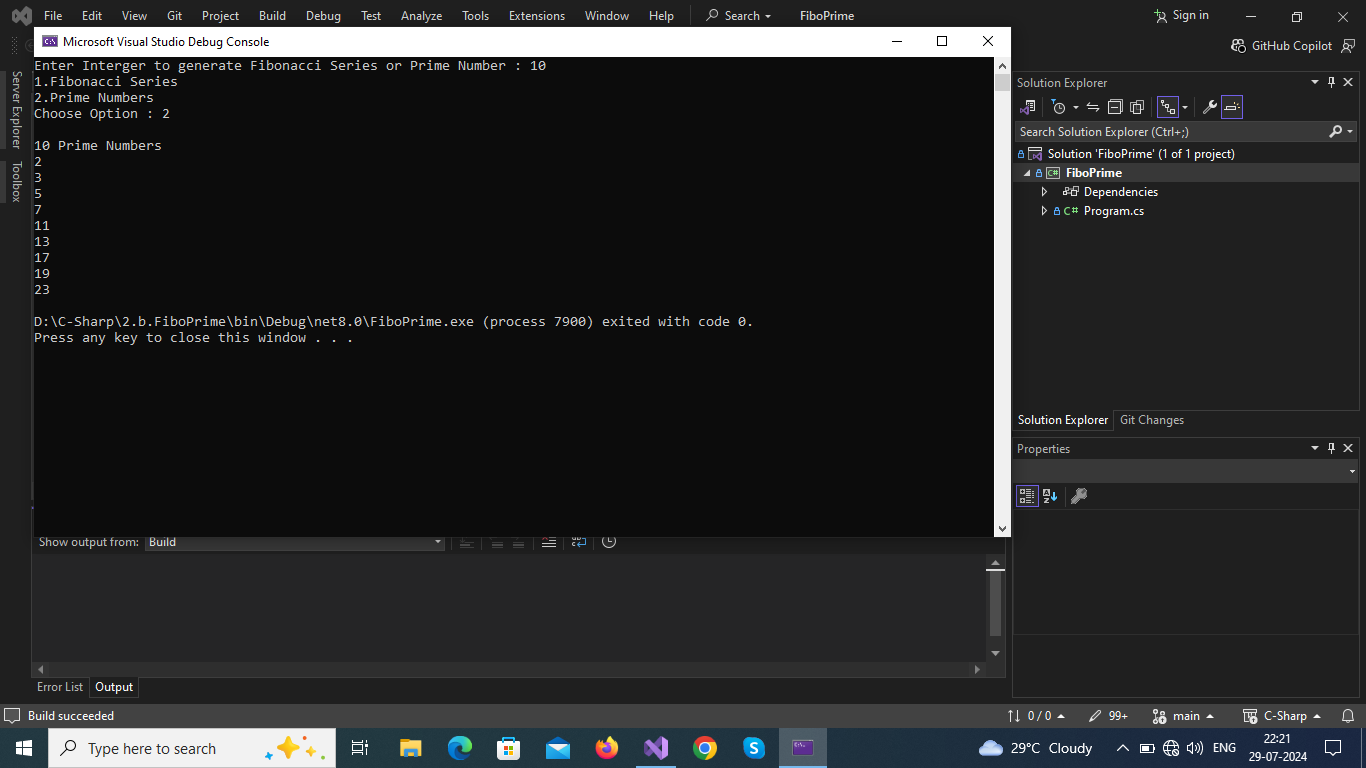
**}**

**}**

**}**

**Output :**





**Result :**

Thus the program executed successfully and ouput was verified.

3.Write a C# program to create a Stack class and appropriate methods to

implement the stack operations.

Aim :

To Write a C# program to create a Stack class and appropriate methods to implement the stack operations.

Algorithm:

1.Start the program

2.Create a class with Stack with its basic function:

1.push(data)

2.pop()

3.peek()

4.Display()

3.initialize stack total size as you own.

4.Make a default constructor to top as -1

5.Push(data):

check stack is full or not

if full:

give message

else:

push a element to stack and increase top by 1

6.Pop():

check stack is empty or not

if empty:

give message

else :

return a element on top and reduce top by 1

7.Peek():

like Pop()

but don't reduce top

8.Display():

use loop to display all elements.

9.Get user input as per need and use switch to select appropriate function.

10.stop the program

Program :

using System;

namespace CS3

{

public class Stack

{

static readonly int MAX = 100;

int[] stack = new int[MAX];

int top;

bool isStackEmpty()

{

return (top < 0);

}

public Stack()

{

top -= 1;

}

internal bool Push(int data)

{

if (top >= MAX)

{

Console.WriteLine("\nStack Overflow");

return false;

}

else

{

stack[++top] = data;

Console.WriteLine("\n{0} --> pushed into stack ", data);

return true;

}

}

internal int Pop()

{

if (isStackEmpty())

{

Console.WriteLine("\nStack Underflow");

return 0;

}

else

{

int value = stack[top--];

return value;

}

}

internal void Peek()

{

if (isStackEmpty ())

{

Console.WriteLine("\nStack Underflow");

return;

}

else

{

Console.WriteLine("\nThe topmost element of the stack - {0} .\n", stack[top]);

return;

}

}

internal void PrintStack()

{

if (isStackEmpty ())

{

Console.WriteLine("\nStack Underflow");

return;

}

else

{

Console.WriteLine("\nElements in Stack are ...");

for (int i = top; i >= 0; i--)

{

Console.WriteLine(stack[i] + "\n");

}

}

}

internal int Menu()

{

Console.WriteLine("\nStack Operations..\n1.Push\n2.Pop\n3.Peek\n4.Allelements\n5.exit");

Console.Write("Enter Choice : ");

return (Int32.Parse(Console.ReadLine()));

}

public static void Main(string[] args)

{

Stack st = new Stack();

int ch = 1;

while (ch != 0)

{

ch = st.Menu();

Console.Clear();

switch (ch)

{

case 1:

Console.Write("\nEnter Data to push into stack : ");

int data = Int32.Parse(Console.ReadLine());

st.Push(data);

break;

case 2:

Console.WriteLine("\n{0} <-- Popped from stack", st.Pop());

break;

case 3:

st.Peek();

break;

case 4:

st.PrintStack();

break;

case 5:

Console.WriteLine("\nThank You ");

ch = 0;

break;

default:

Main(args);

break;

}

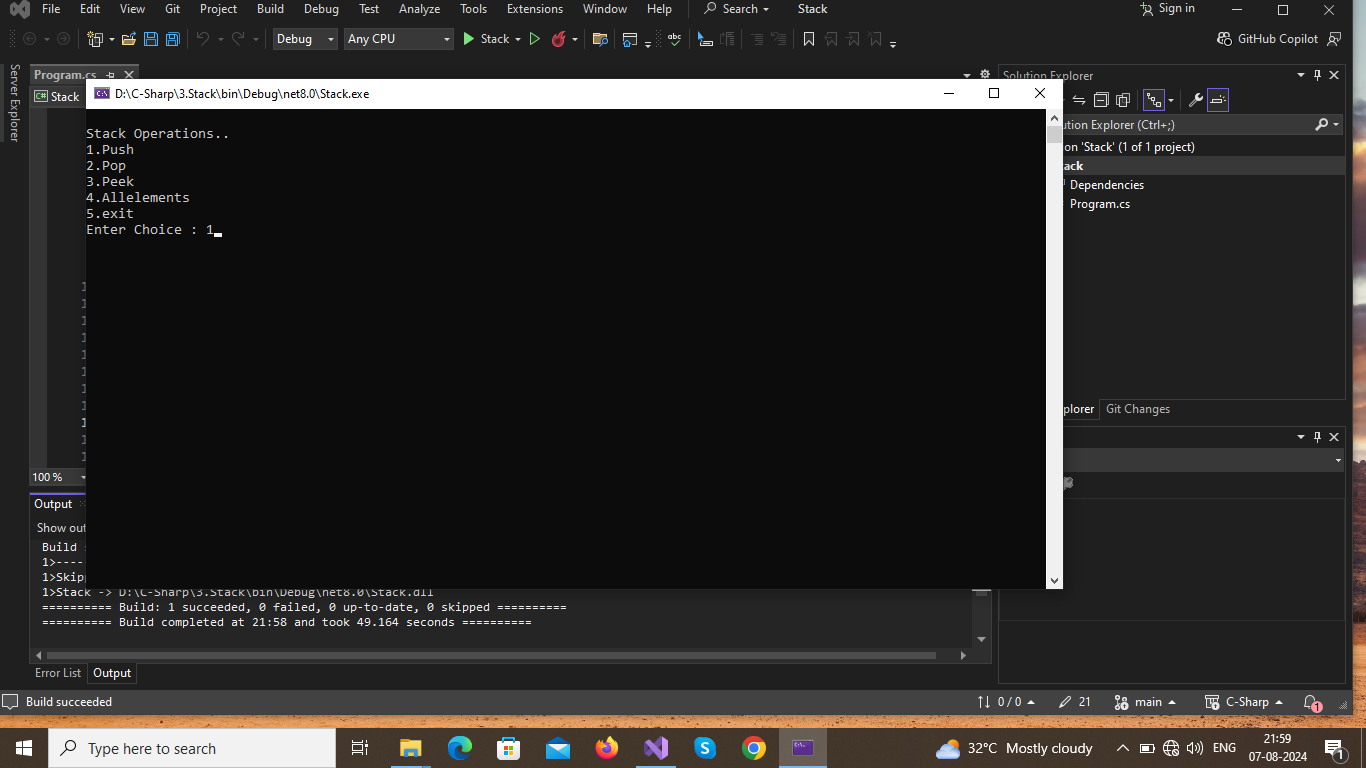
}

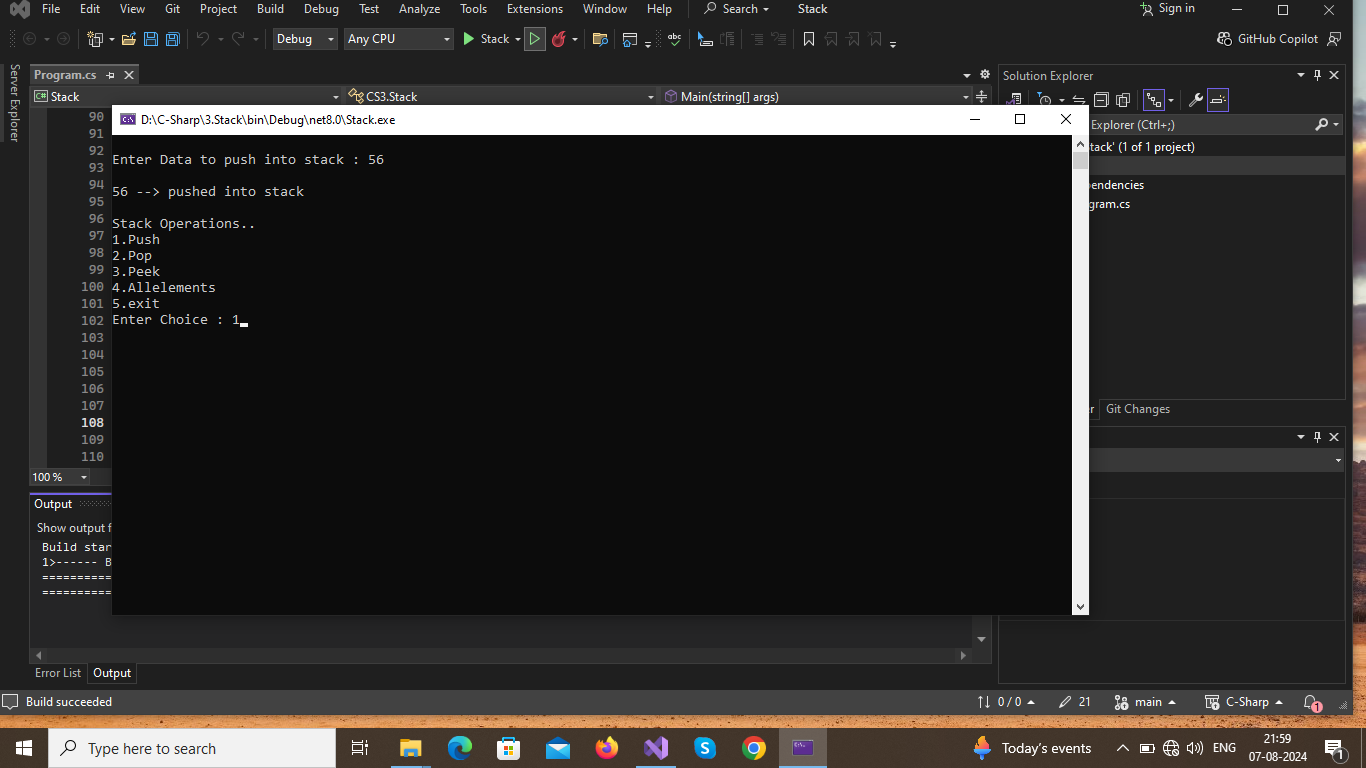
}

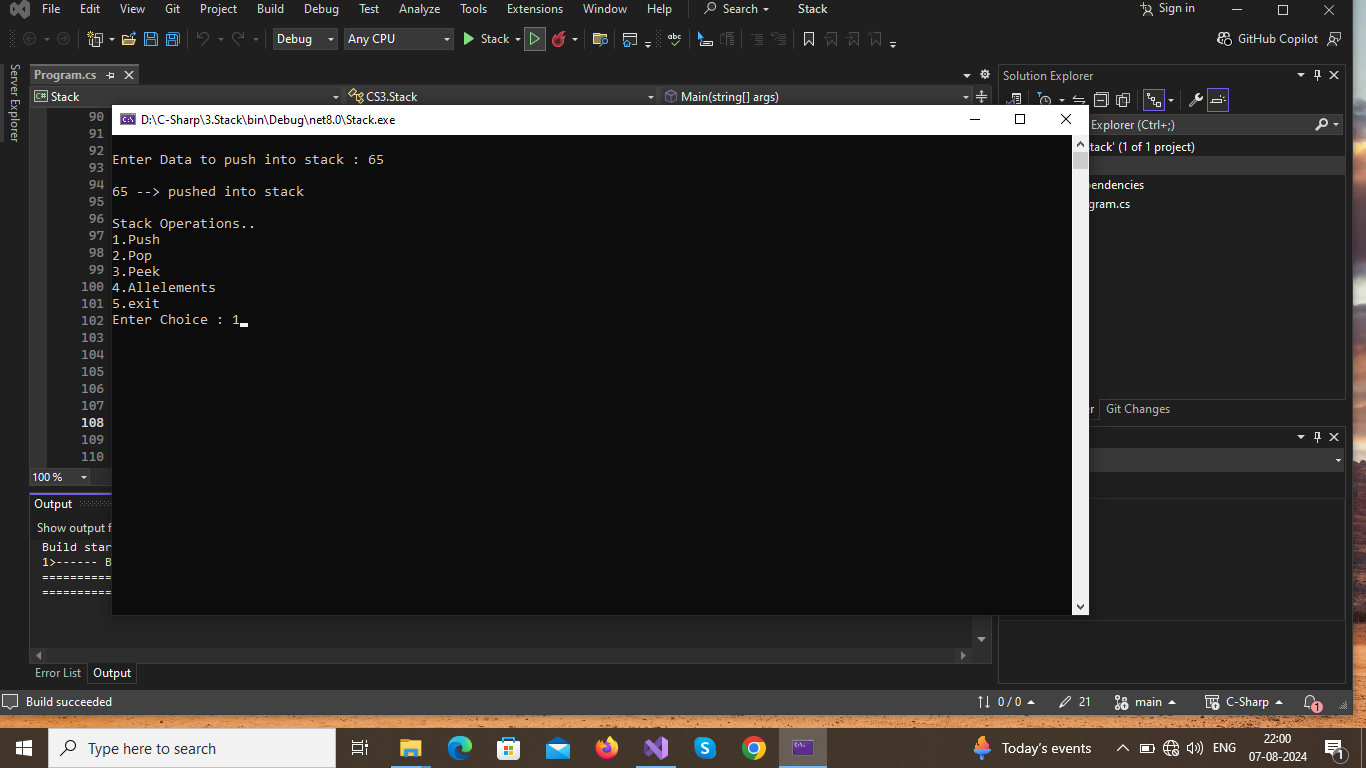
}

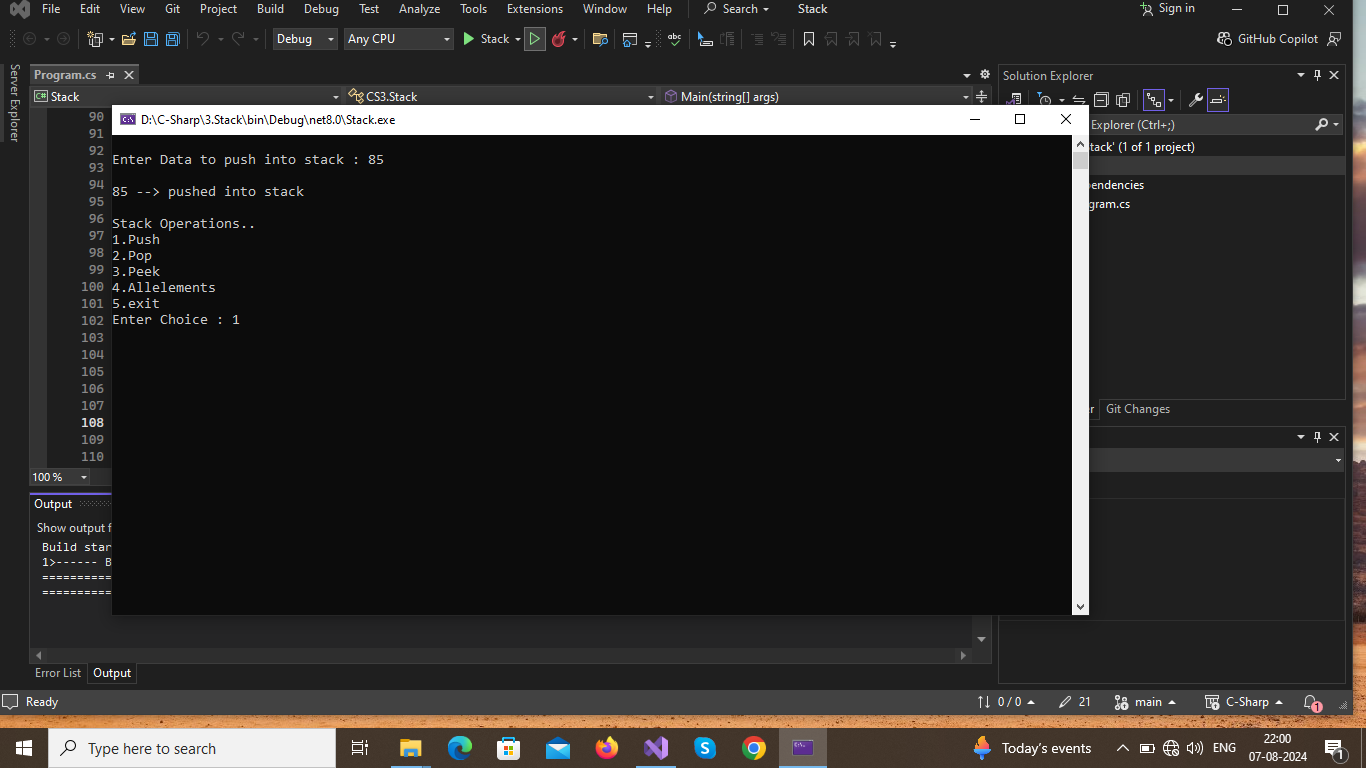
}

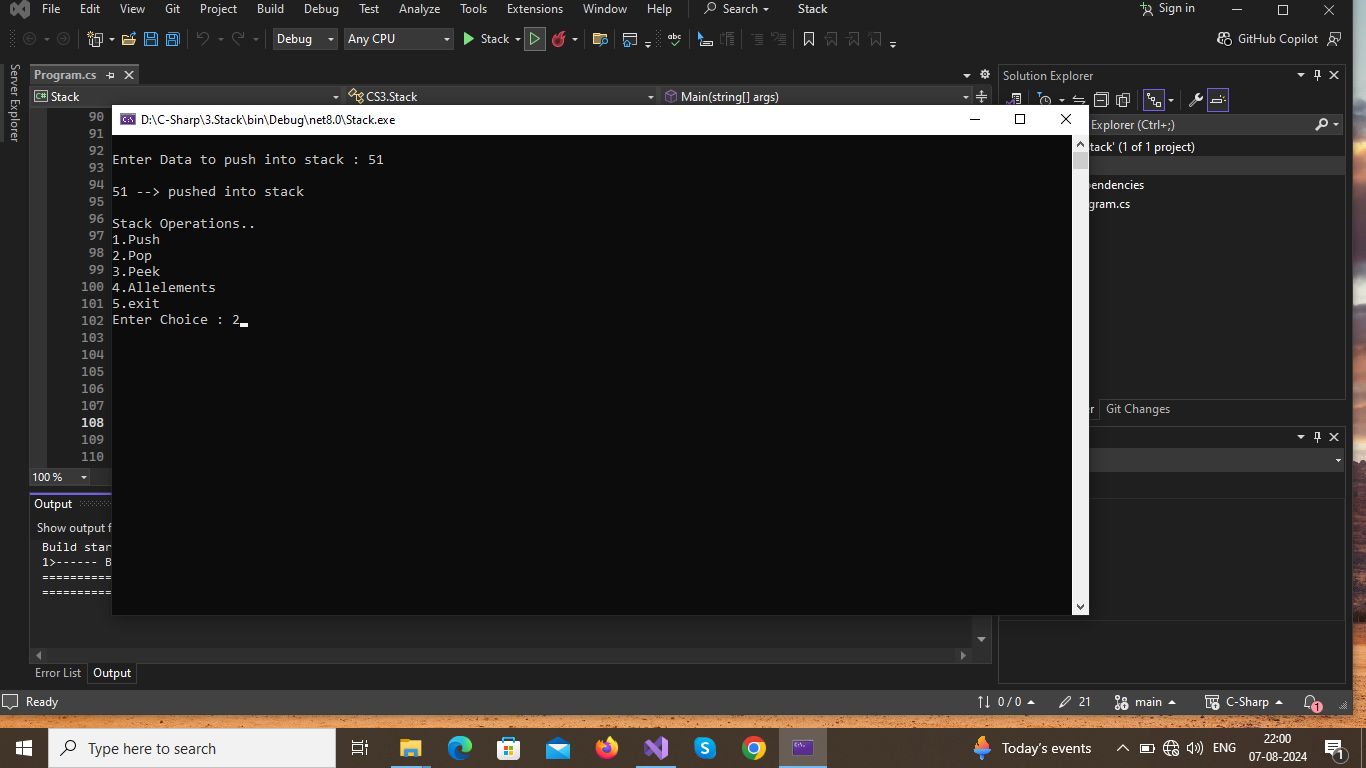
Output :











Write a C# program to create a Stack class and appropriate methods to implement the stack operations.

Store a string "123456789" in a string variable and use it to display thefollowing pattern.

1

232

4

34543

5

6

7

8

9

4567654

567898765

Develop a C# program with more than one class and illustrates the concept of constructor overloading and method overloading.

(a) Demonstrate use of virtual and override key words in C# with a simpleprogram

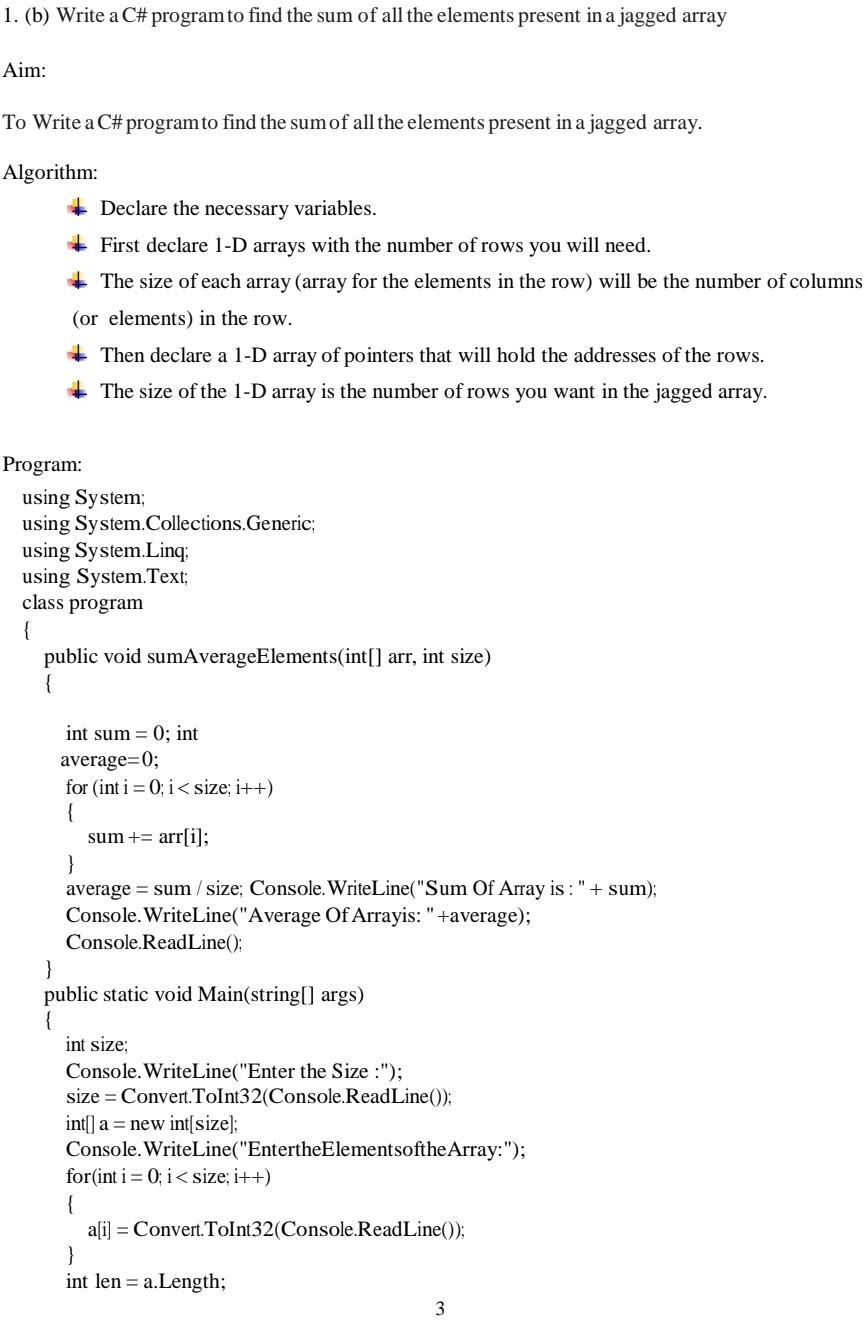
(b) Write a program to demonstrate abstract class and abstract methods inC#.

Design a C# interface for Queue data structure. Develop a class that implements this interface using array. Provide necessary exception handling in the implementations.

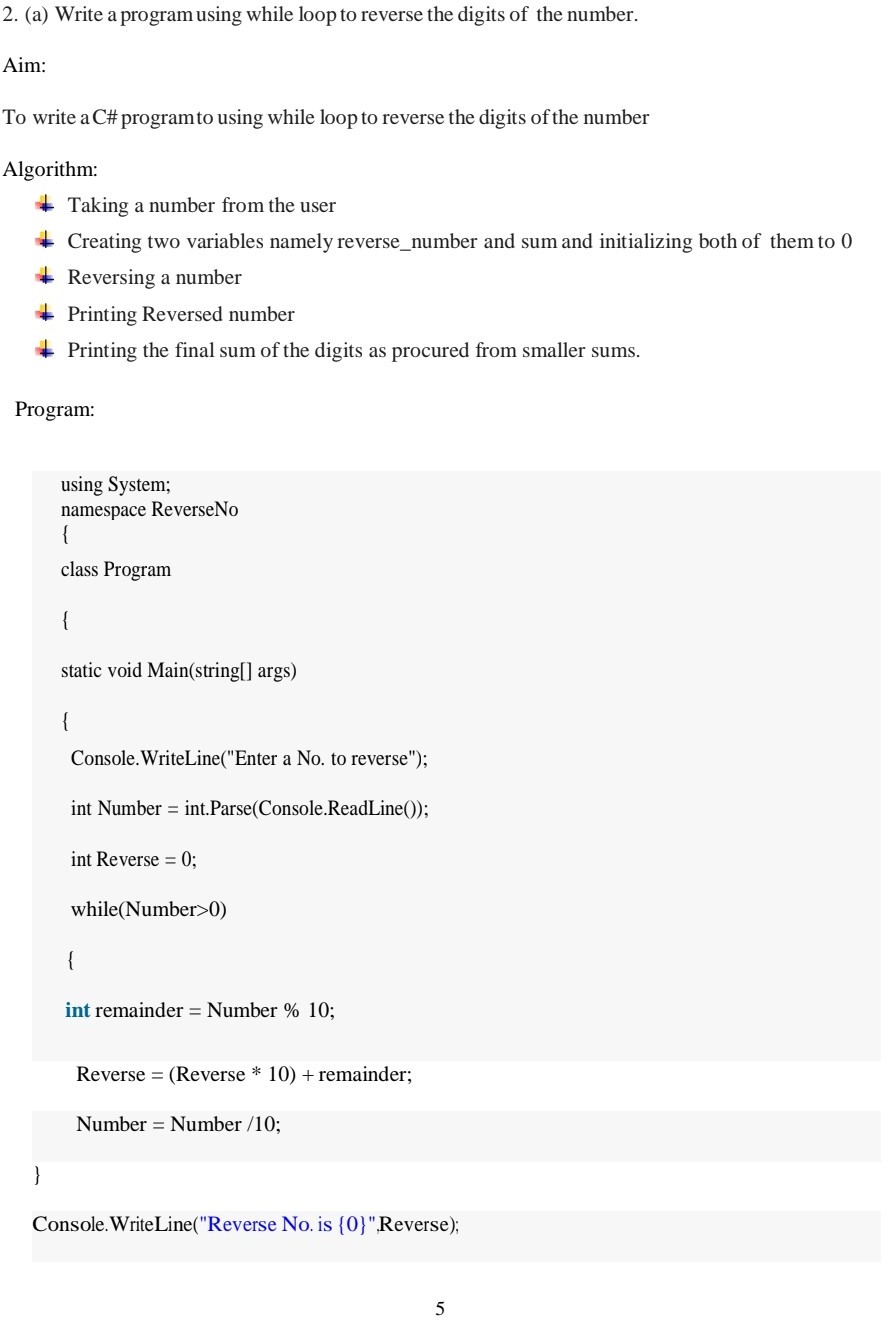
Write a C# program which declares two properties: name and gender in an interface, and provide its implementation in a class which checks for gender and greets the user.

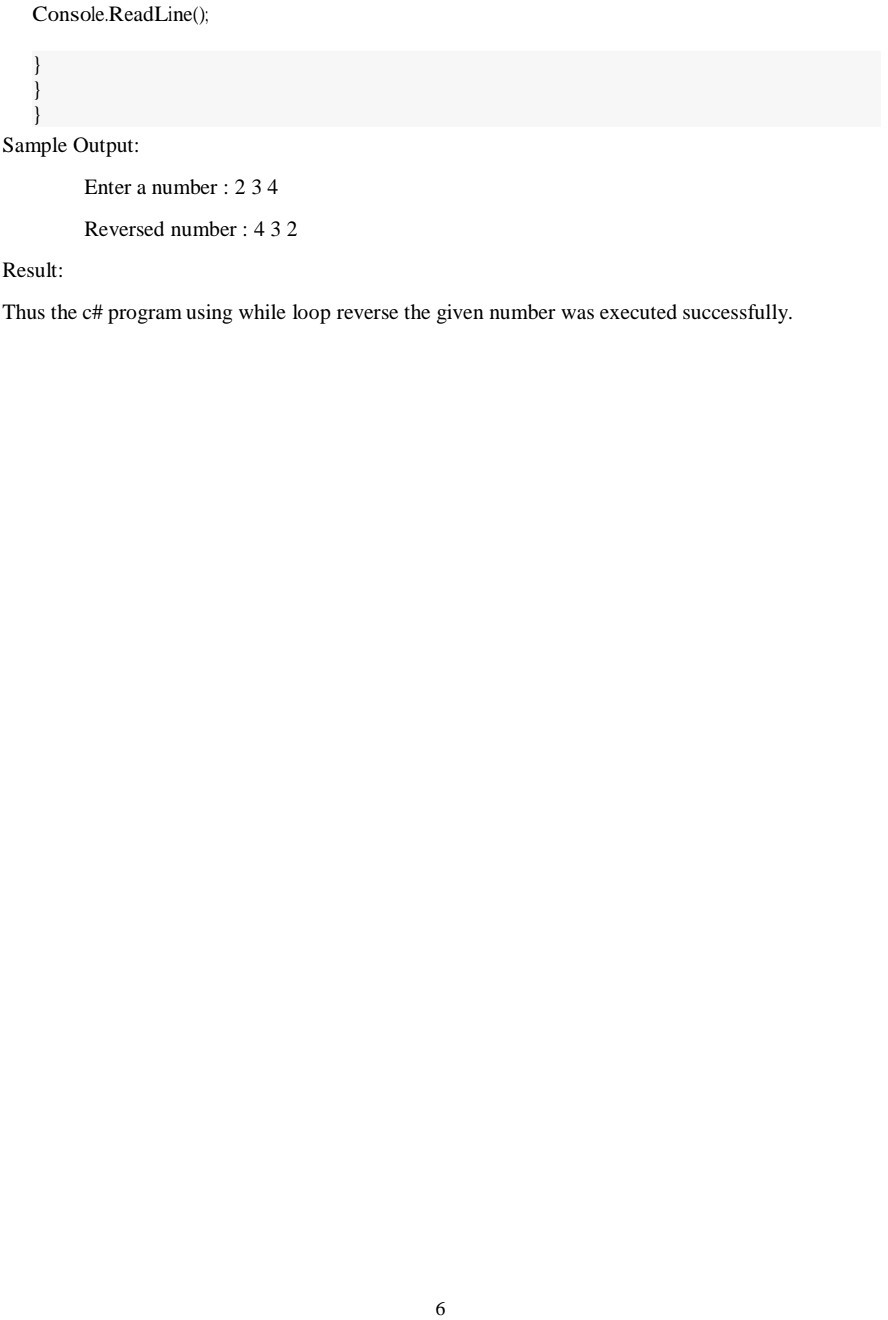
Develop a program that is likely to throw multiple exceptions that are handled using catch and finally blocks. As a part, the program must read a name from the keyboard and display it on the screen. The program should throw an exception when the length of the name is more than 15 characters. Design your own exception handling mechanism.

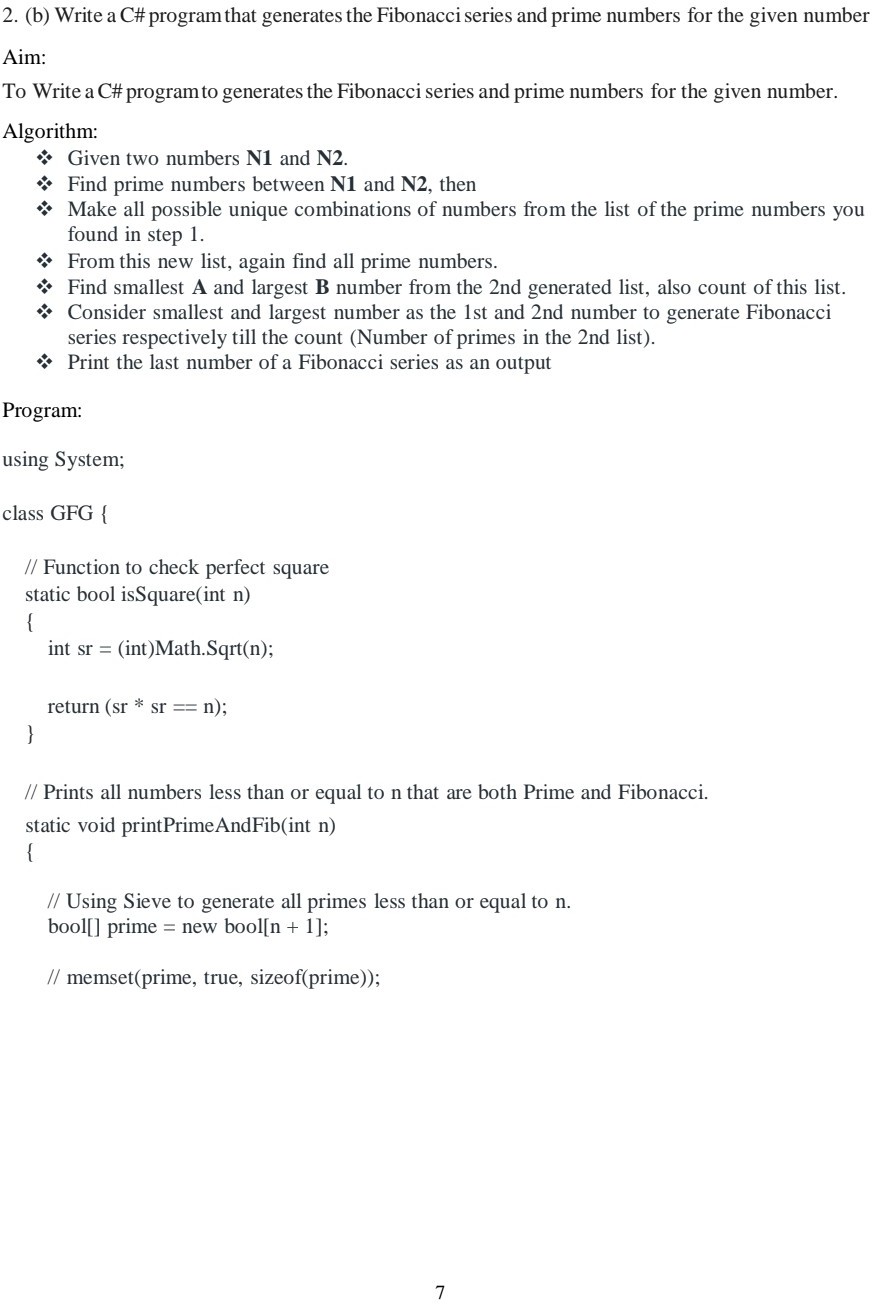
PAGE NO

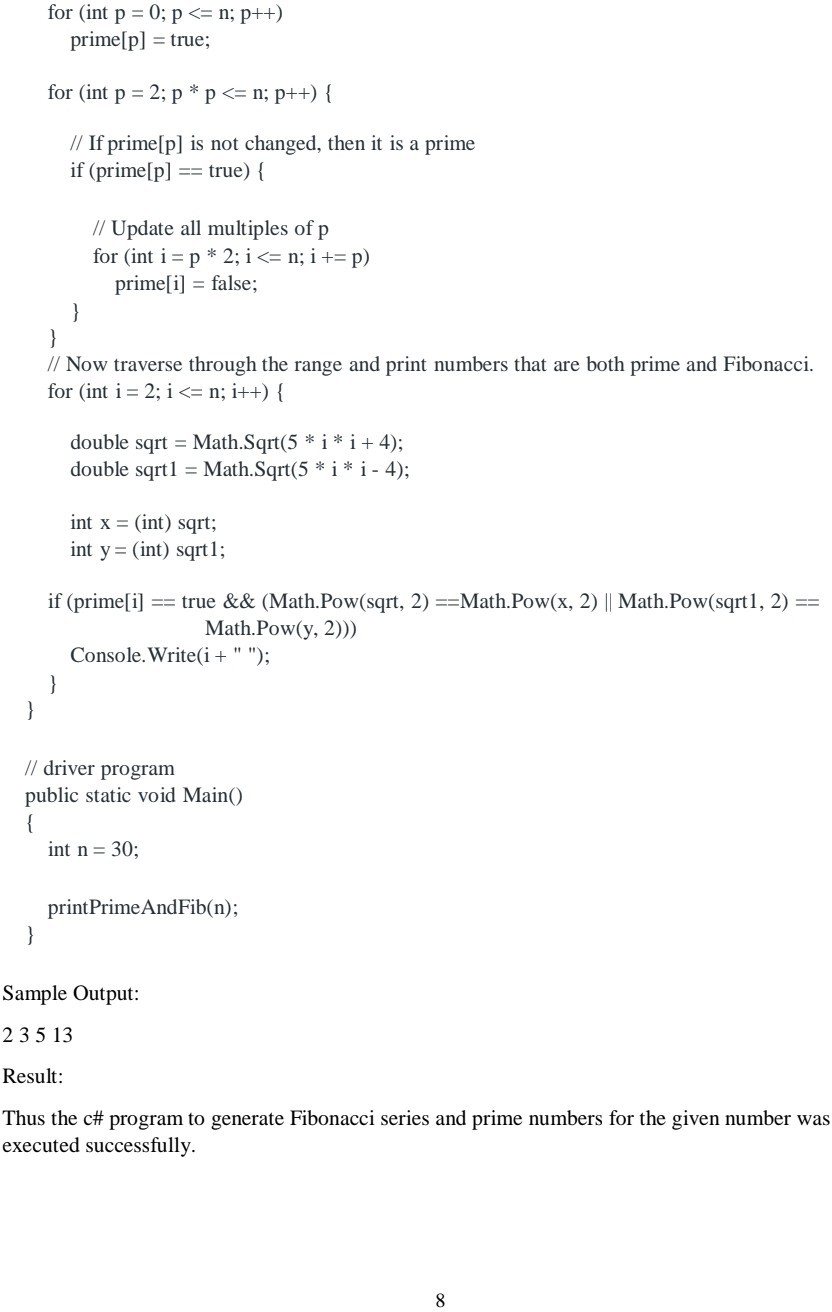


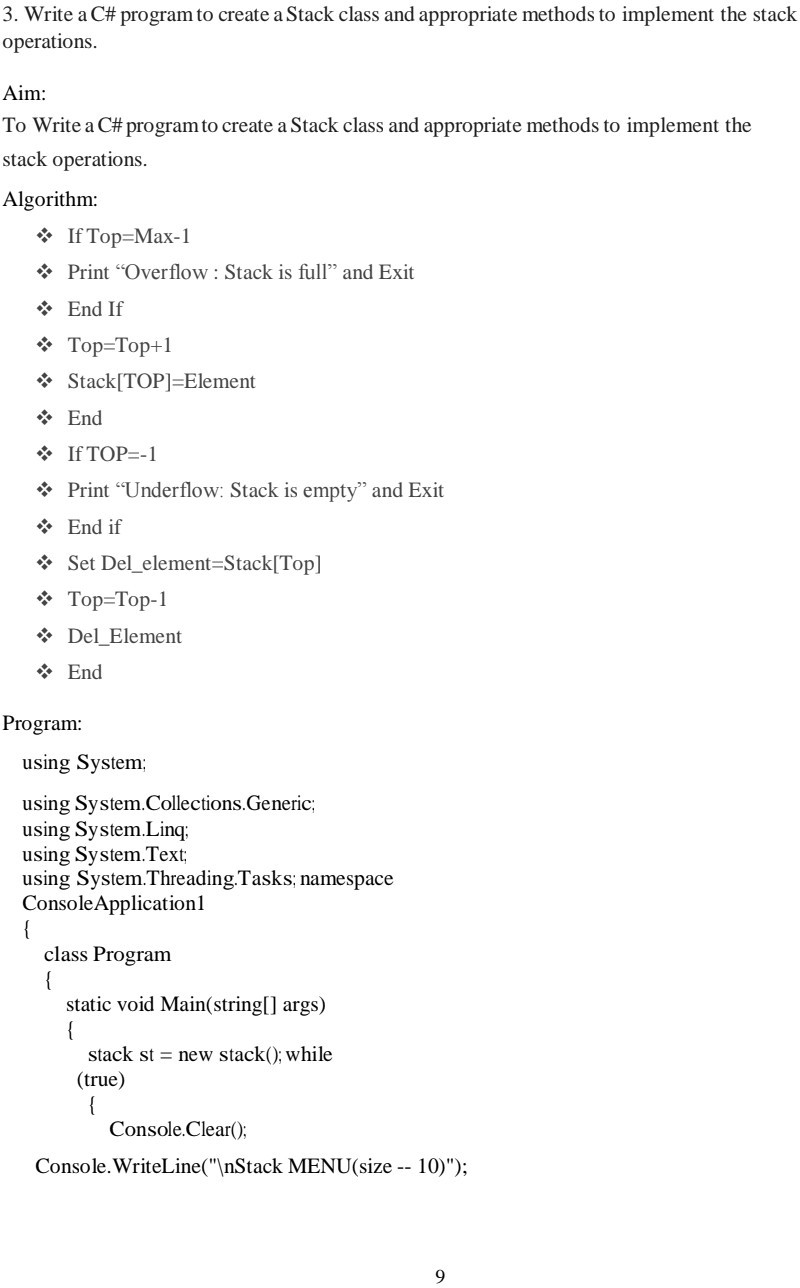


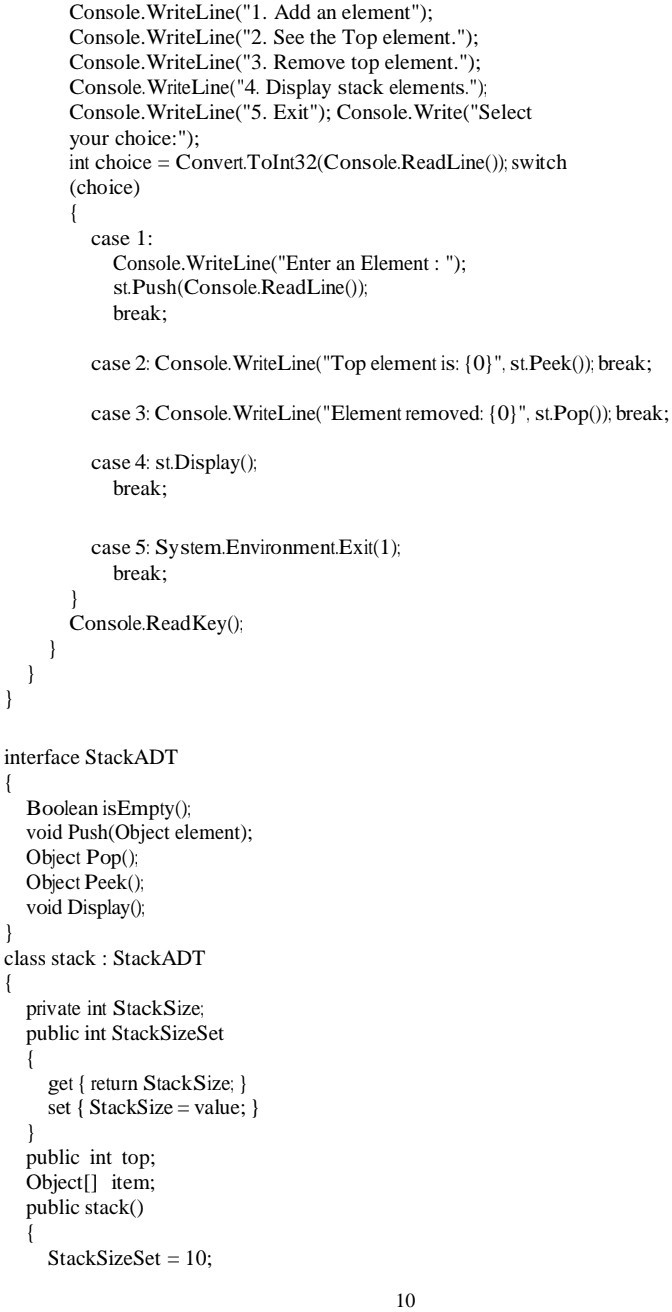


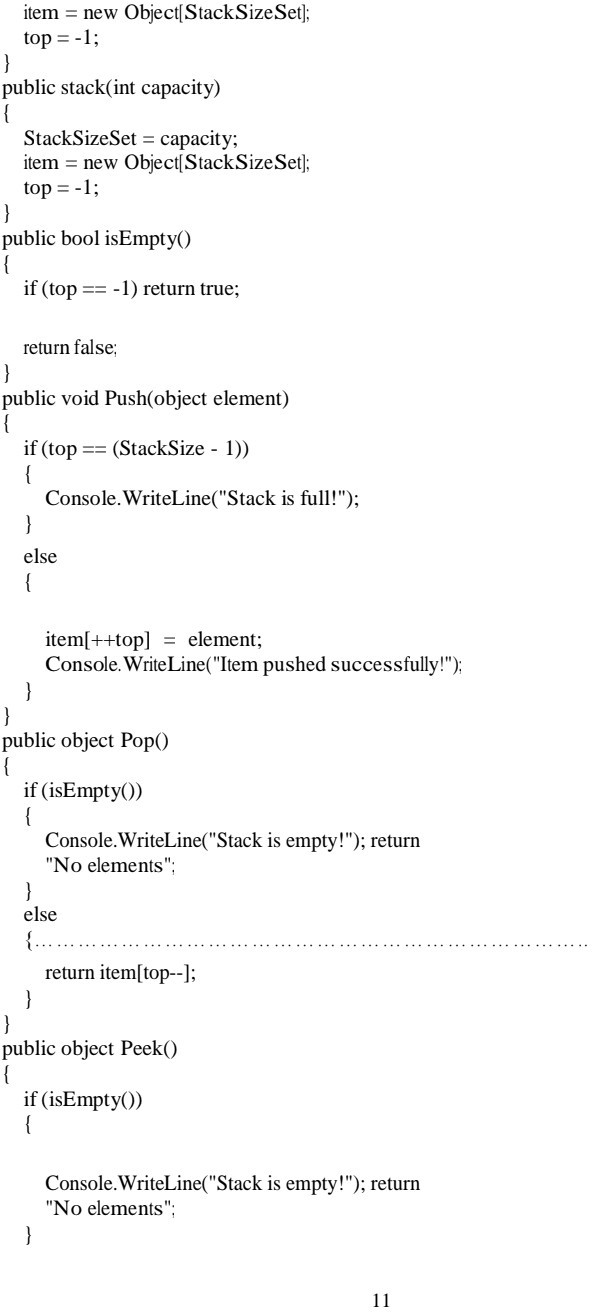




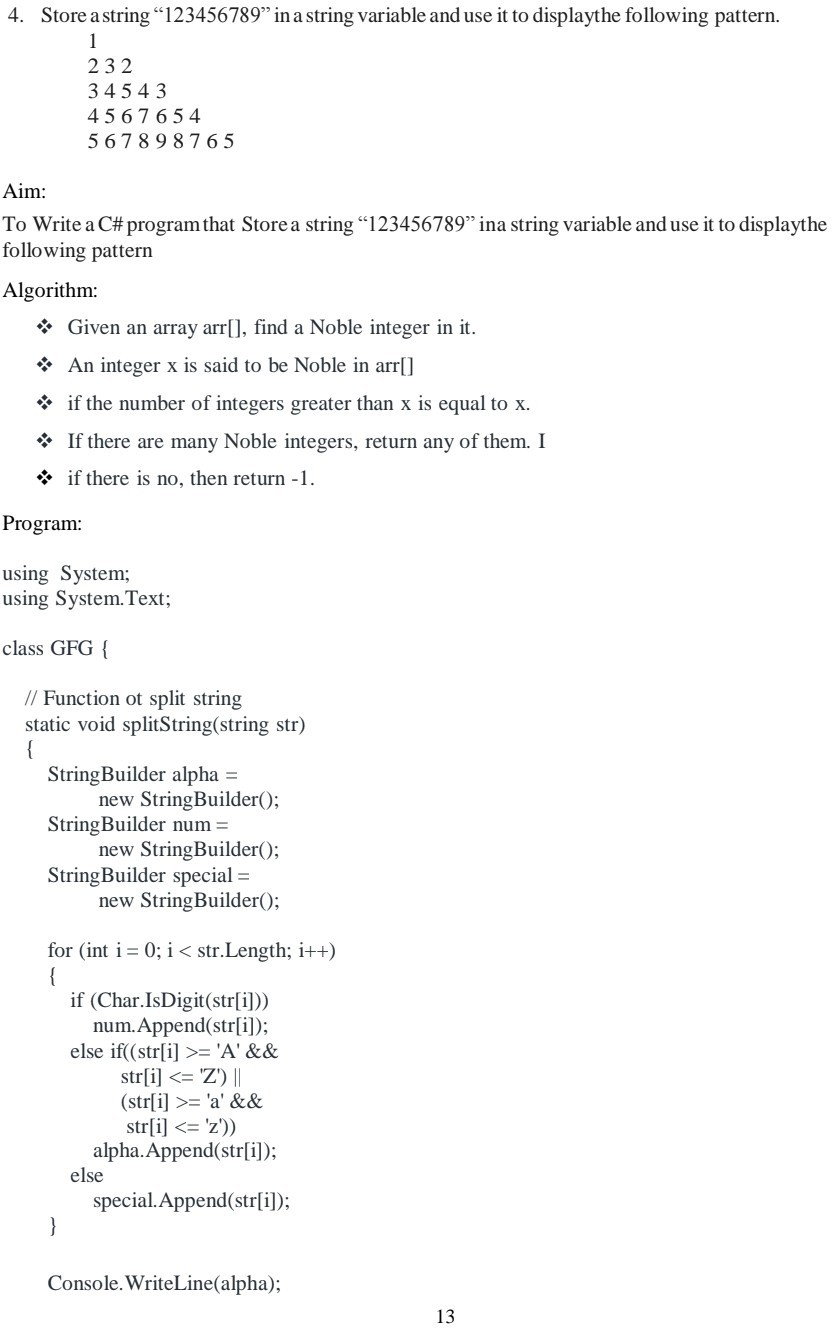


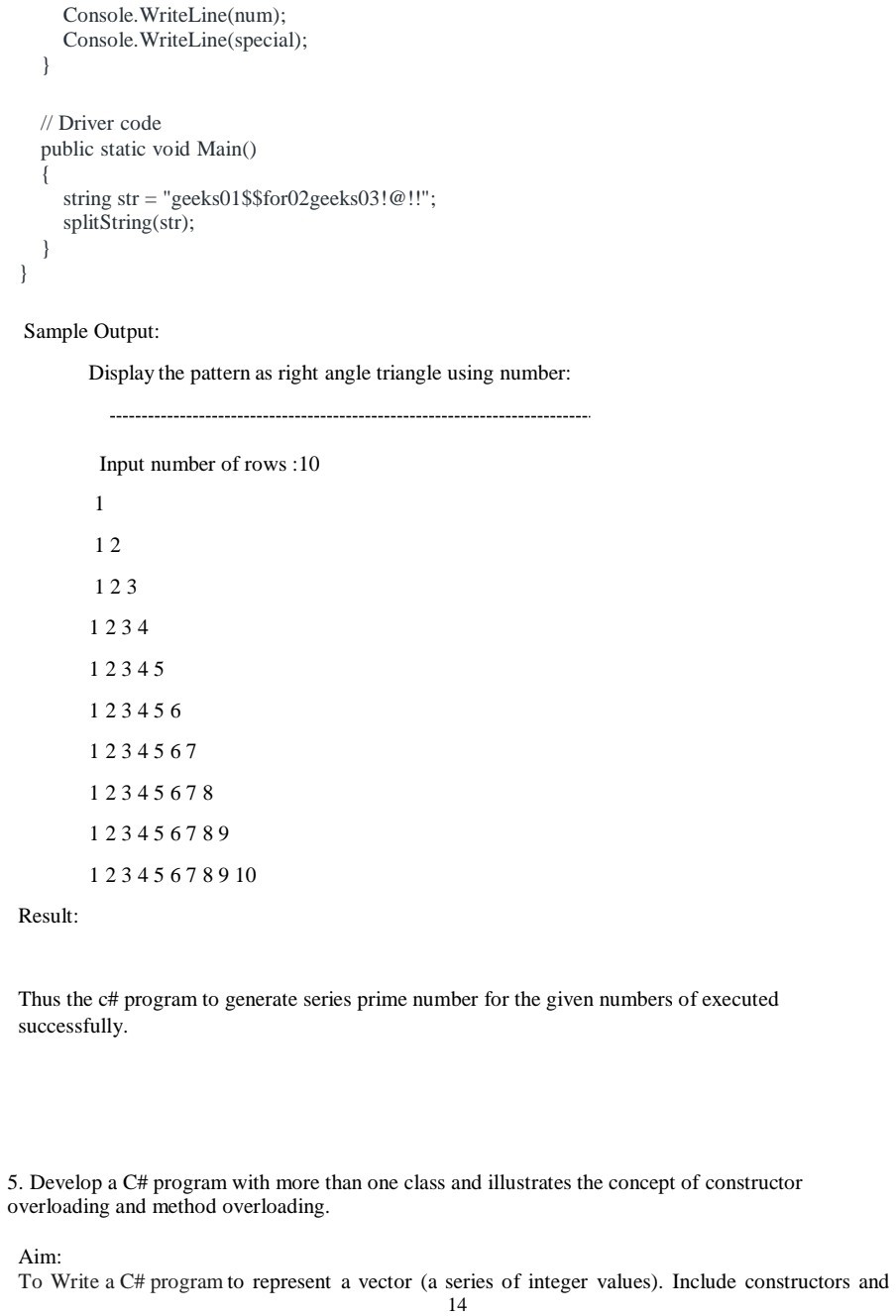


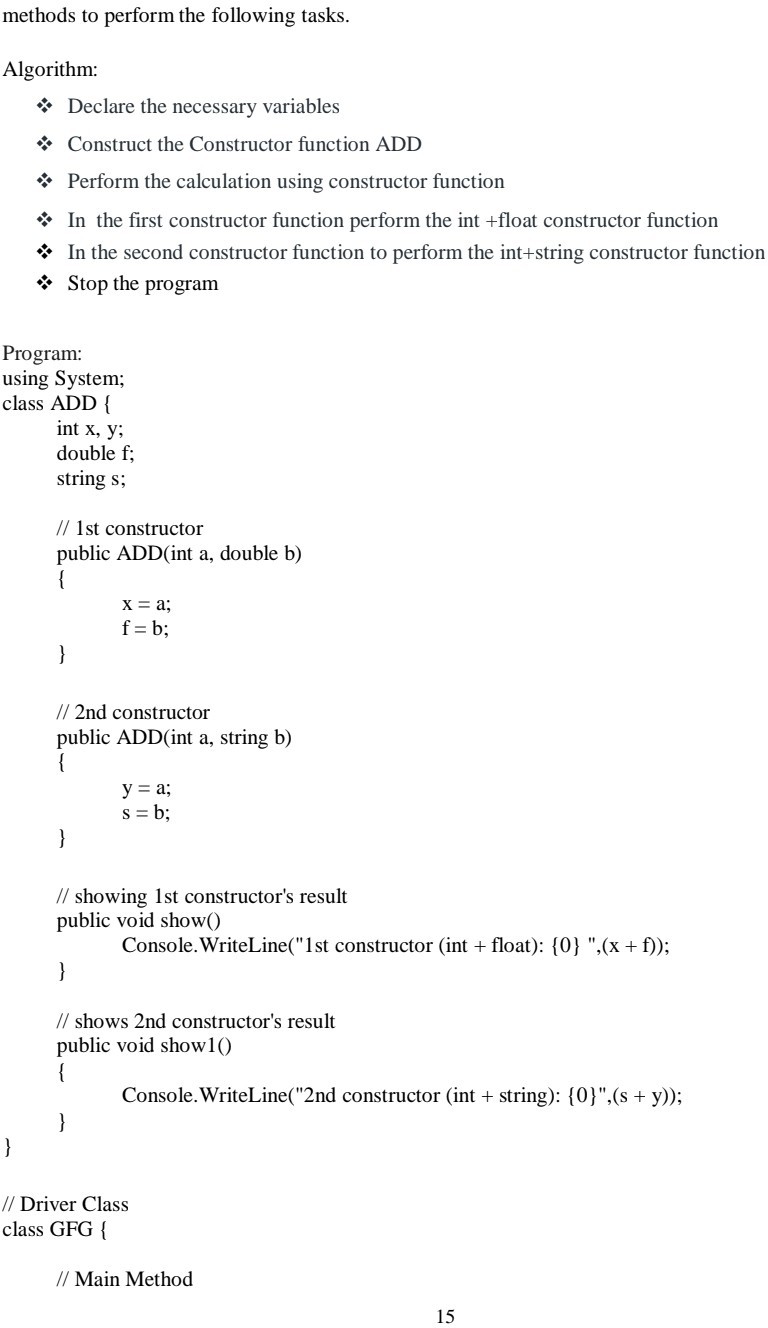


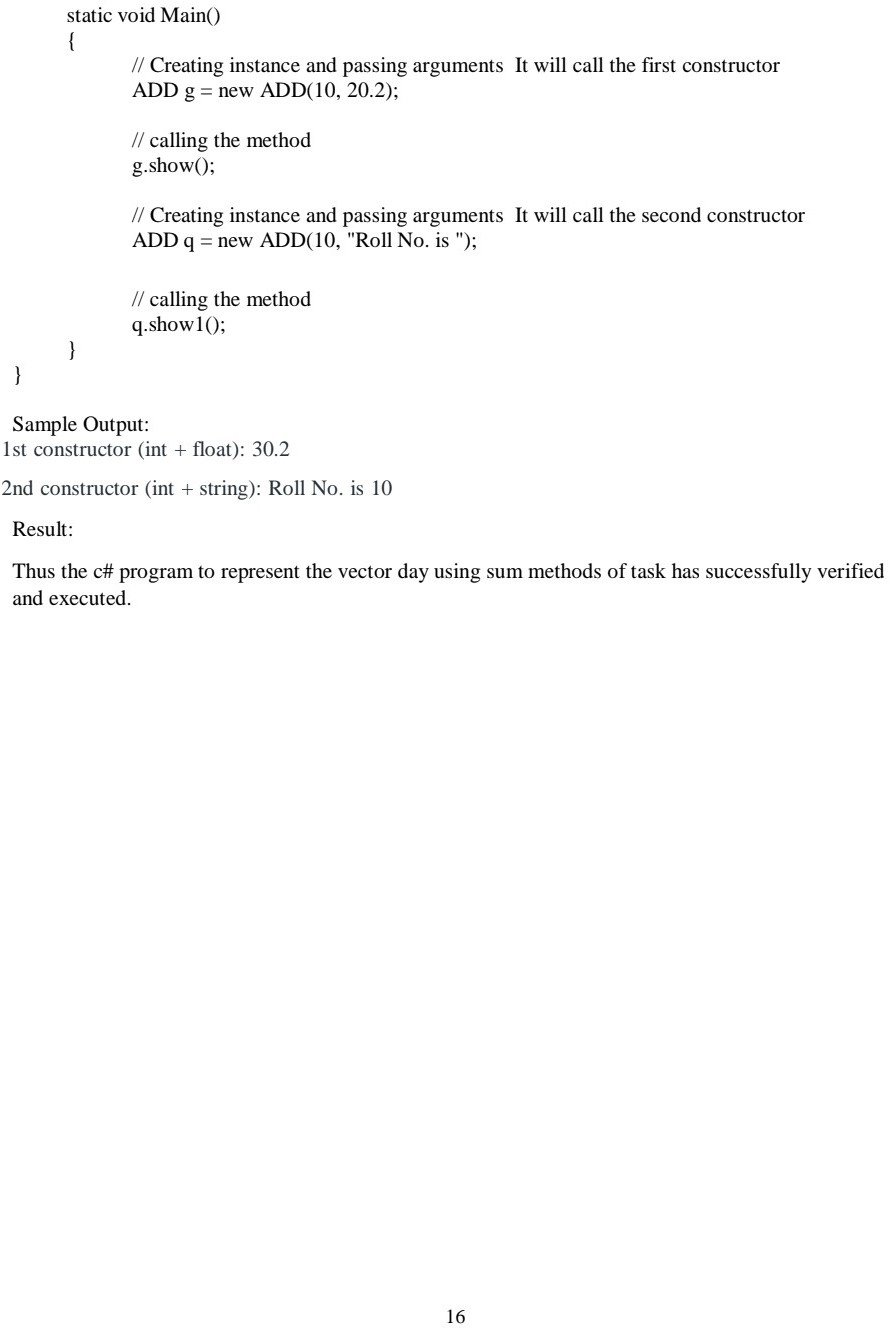


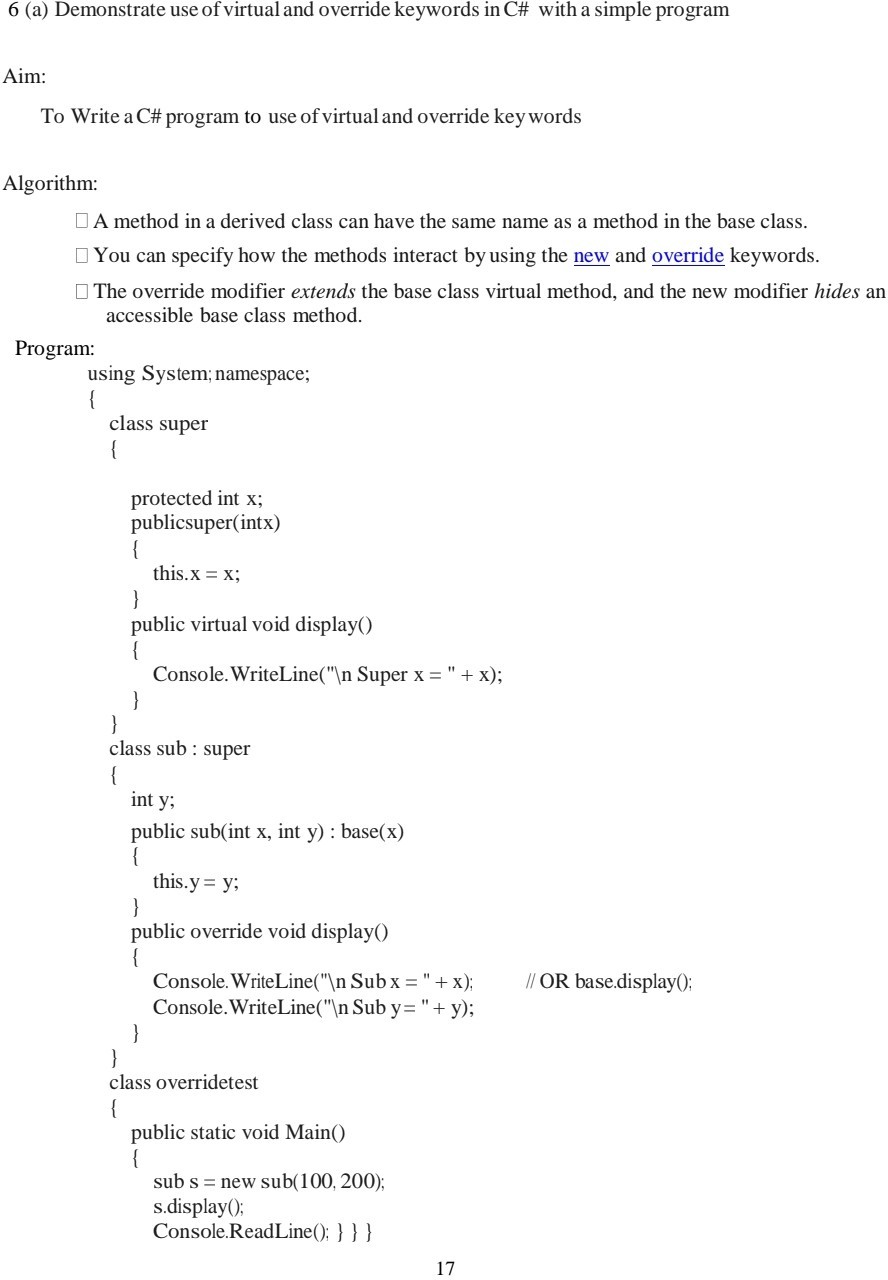


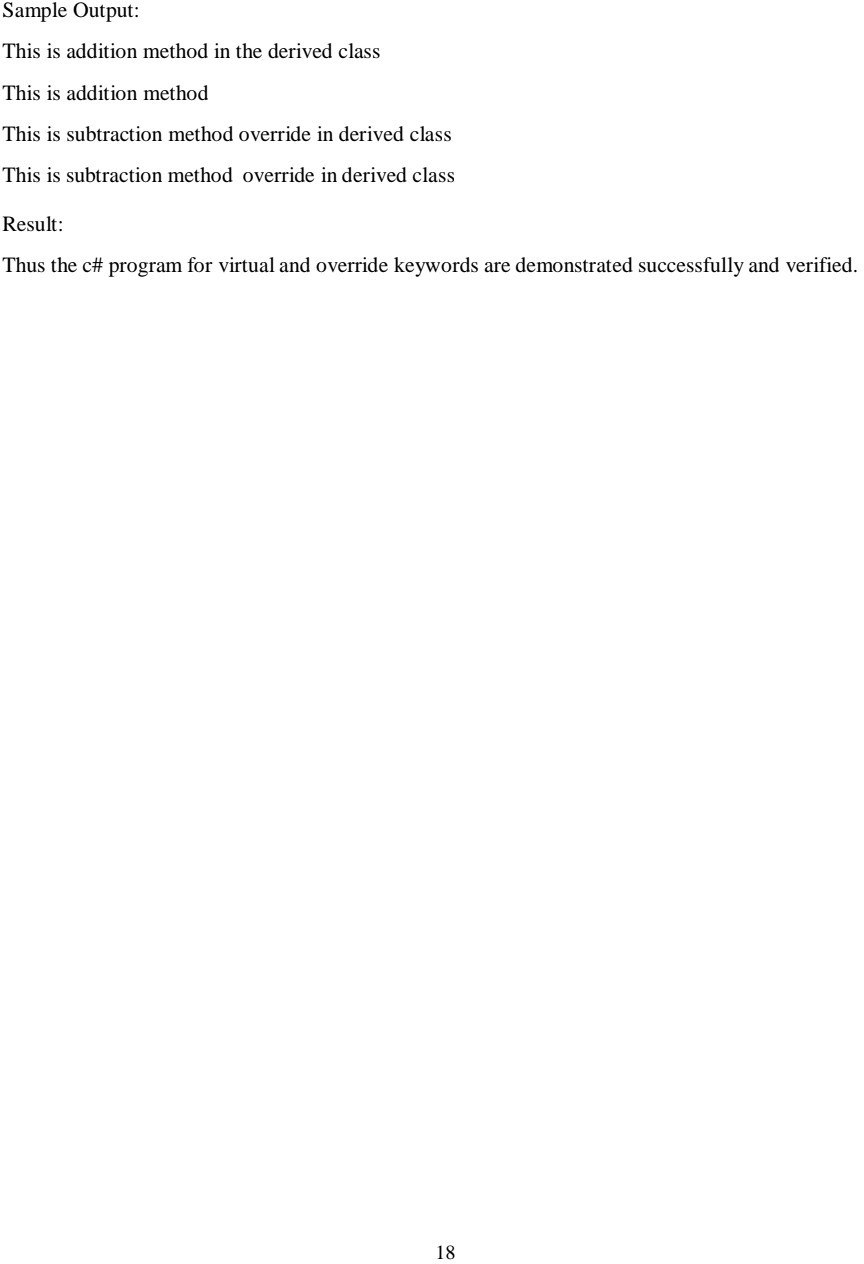


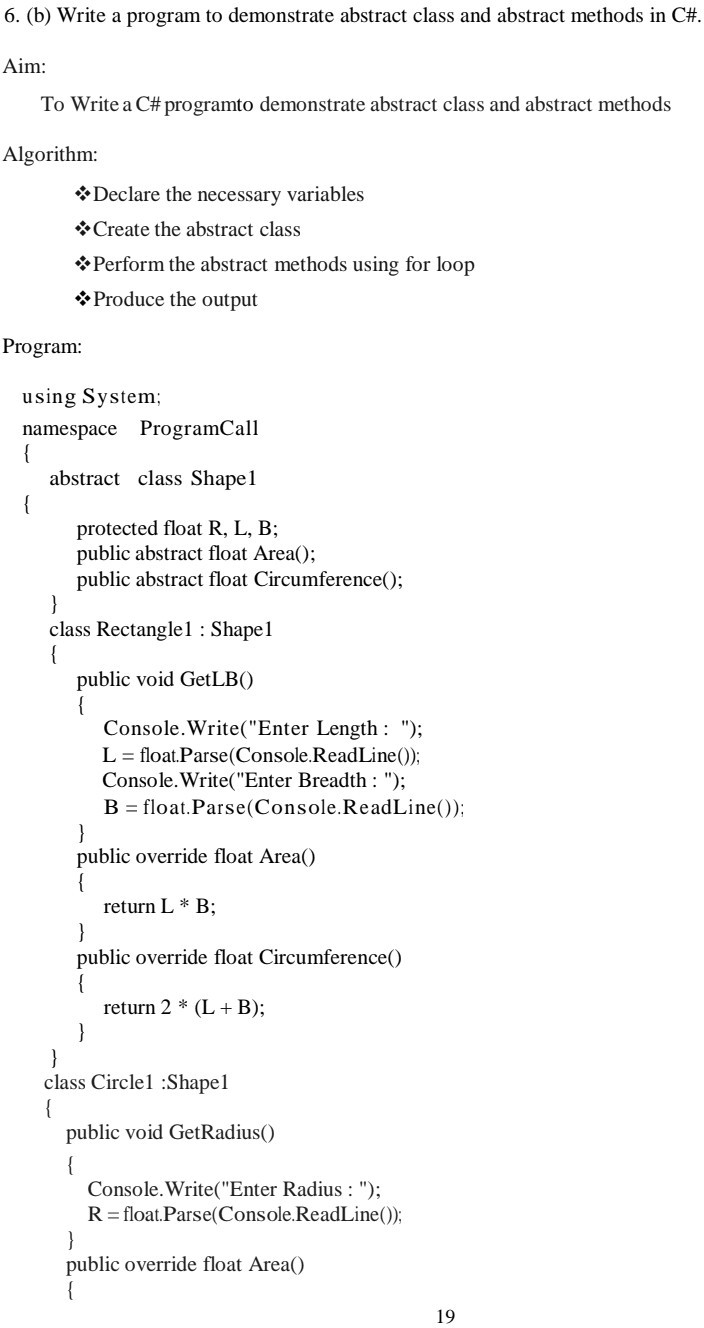


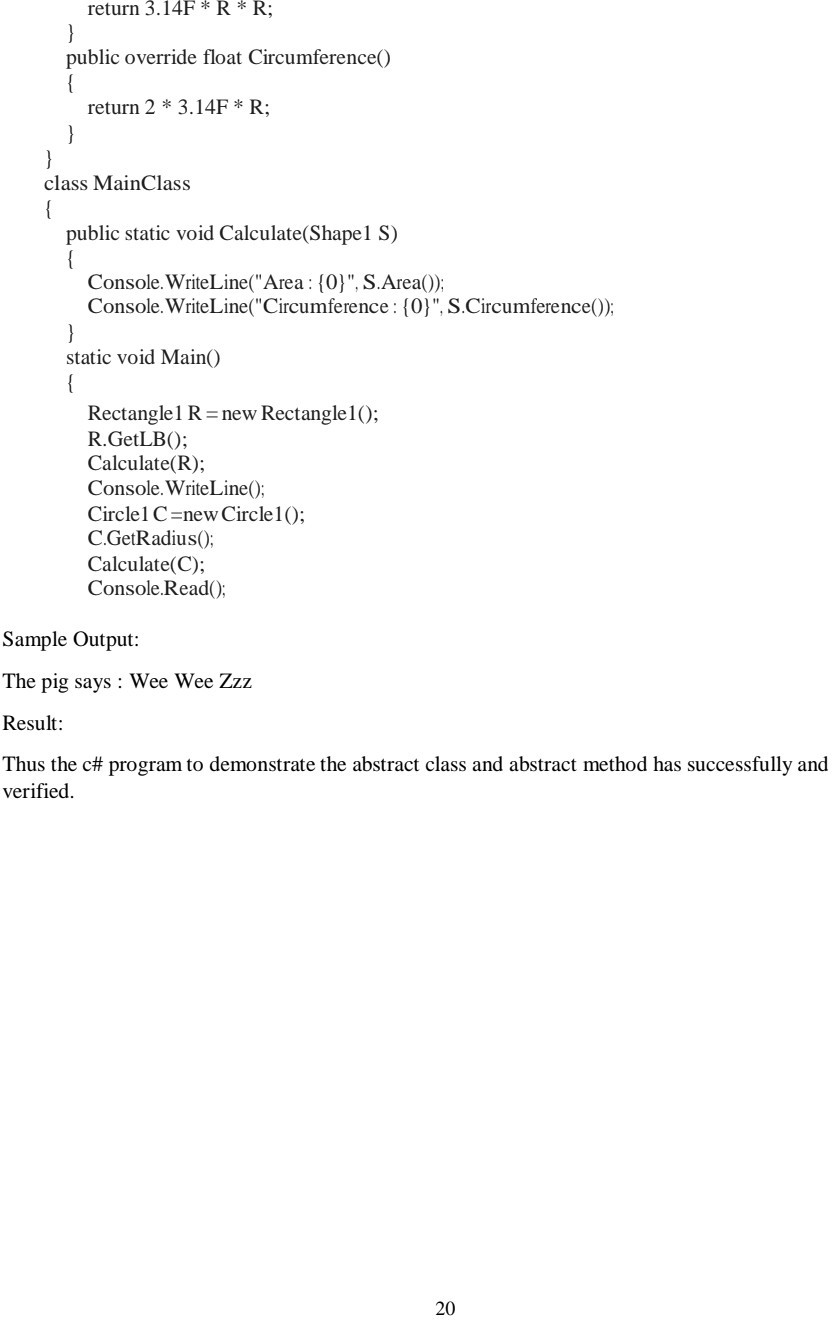


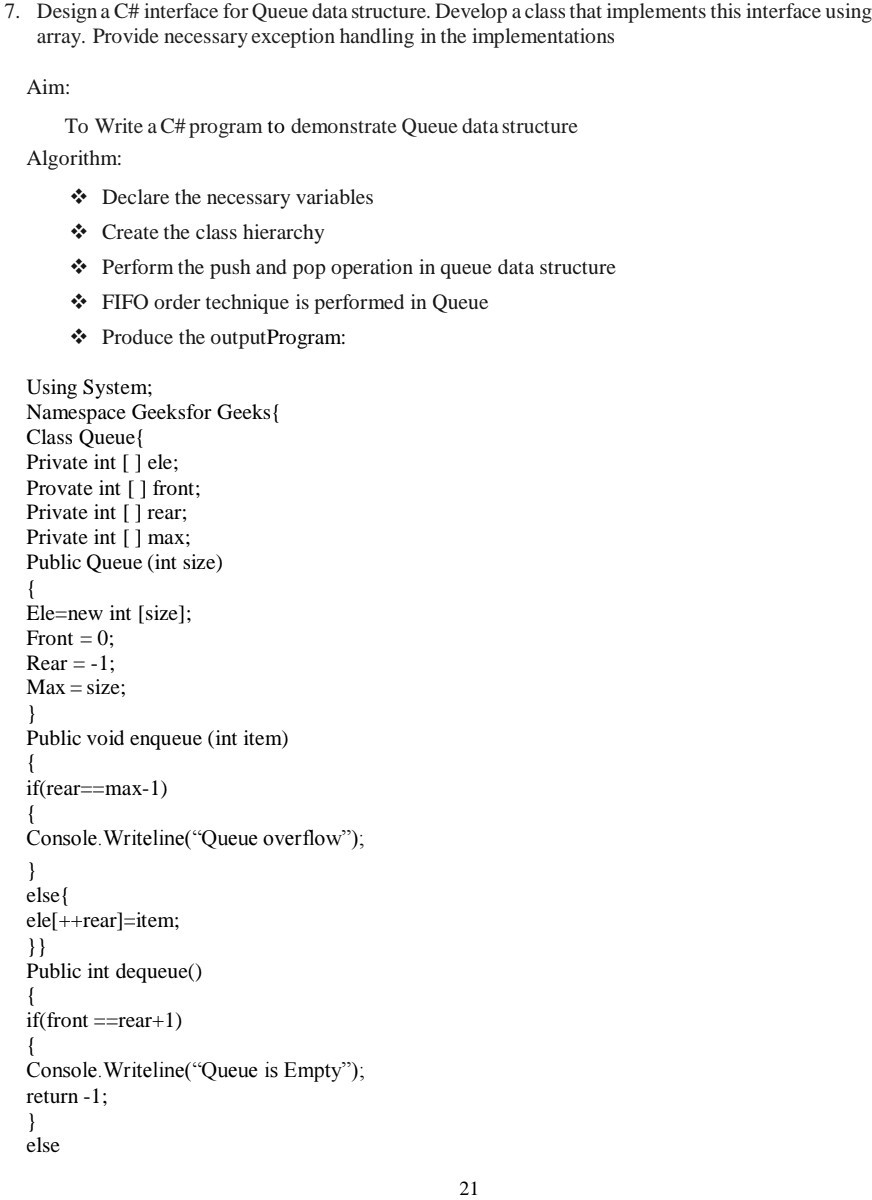


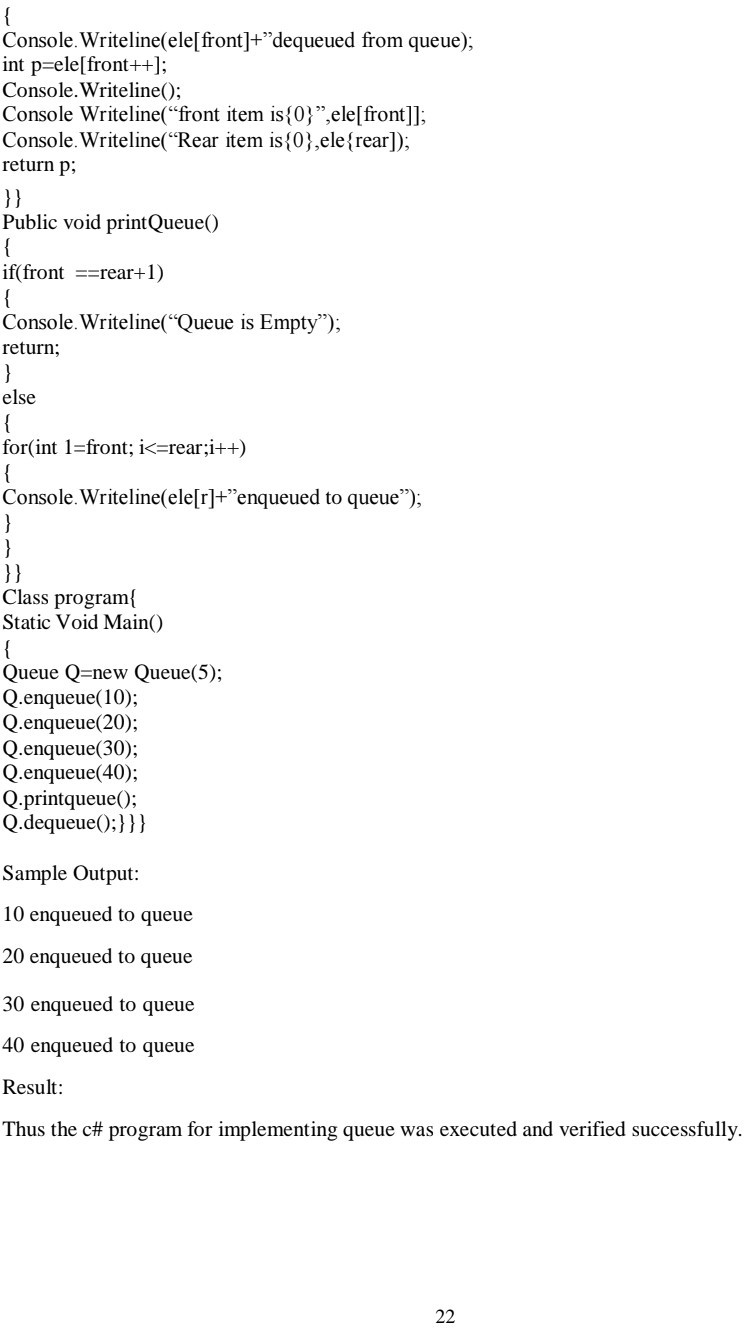


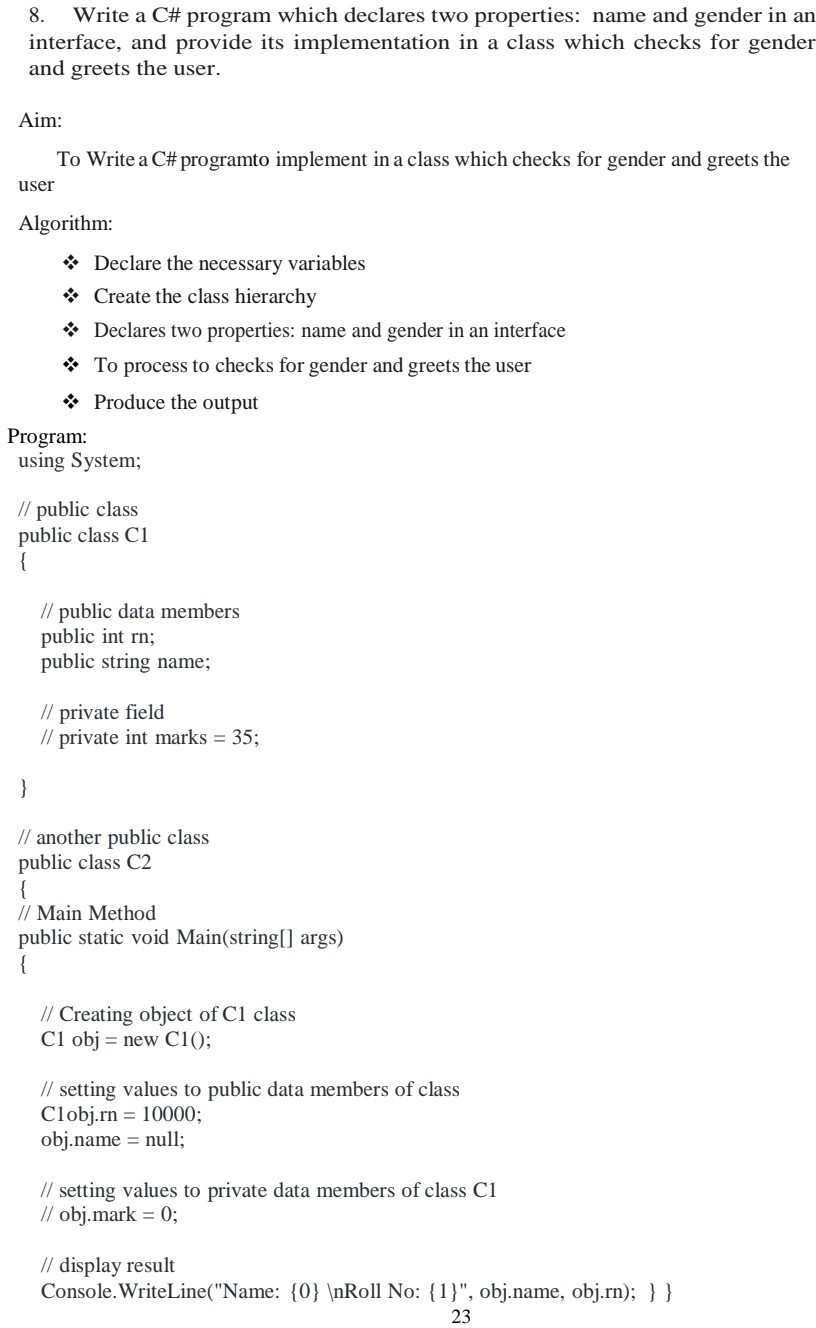












3.Stack

4.String Pattern

1.Start The Program

2.Store value as string in variable as S="123456789"

3.initialize 3 variables eg)i,j,k as 0 (i=0,j=0,k=0)

4.Run two loop to iterate over row and column

5.First loop should run for half of string length(row loop)

6.Second loop should run for twice of the string length(columb loop)

7.print value of string based on below condition:

if(col<=2\*row):

print value of string

to get value(s[j]-'0')

store value in k

else:

--k

if(k>row):

print(k)

8.insert new line after each row completion

9.stop the program

5.

7.Queue Implementation

1.Start Program

2.create an interface that consist of basic queue operations likely

1.Enque()

2.Deque()

3.Front()

4.Rear()

5.Display()

3.Implement using a class (eg:QueueImp)

4.Enqueue() to add element in rear of queue data structure

5.Dequeue() to remove element in front of queue data structure

6.Front() to see element in front of queue ds

7.Rear() to see element in rear of queue ds

8.Display() to see all elements in a queue ds.

9.Stop the program

8.Interface to Greet.

1.Start program

2.create an interface that consist method with parametera as name and gender

3.create an implementation class to implement an interface method to get data such as name and gender.

4.Print a greeting message based on gender to welcome user.

5. Stop the Program

9.Exception Handling

1.Start the program

2.Create a try catch block to illustrate exception handling

3.Get a Name as input in try block and check the name has less than or equal to 15 characters

4.if it is more than 15 then raise an exception and give a message

5.add more exceptions like division by 0,formatexception,and so on.

6.stop the program