**QUESTION 1: Write a program to build your own stack class**

**Code:**

**Program Class:**

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Enter the length of your stack");

int Length = Convert.ToInt32(Console.ReadLine());

Stack obj = new Stack(Length);

while (true)

{

Console.WriteLine("You want to perform any function on your stack");

Console.WriteLine("1 - Push\n2 - Pop \n3 - Peek ");

int answer = Convert.ToInt32(Console.ReadLine());

if (answer == 1)

{

Console.WriteLine("Enter value to push in stack");

int val = Convert.ToInt32(Console.ReadLine());

if (obj.Push(val))

{

Console.WriteLine("Value successfully pushed");

obj.Display();

}

else

{

Console.WriteLine("Stack overflow");

}

}

else if (answer == 2)

{

int val = obj.Pop();

if (val >= 0)

{

Console.WriteLine("Value successfully popped");

Console.WriteLine("Popped value = " + val);

}

else

{

Console.WriteLine("Stack Underflow");

}

}

else if (answer == 3)

{

int val = obj.Peek();

if (val >= 0)

{

Console.WriteLine("Top value = " + val);

}

else

{

Console.WriteLine("Stack empty");

}

}

else

{

break;

}

}

}

**Stack Class:**

class Stack

{

private int Length;

private int top=0;

private int[] array;

public Stack(int length)

{

Length = length;

array = new int[length];

}

public void Display()

{

Console.WriteLine("Your array");

Console.WriteLine();

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

{

Console.Write(array[i]+" ");

}

Console.WriteLine();

}

public bool Push(int val)

{

if (!Overflow())

{

array[Count(this.top)]=val;

return true;

}

return false;

}

public int Count(int top)

{

return this.top++;

}

public int Pop()

{

if(top>0)

{

return array[--this.top];

}

return -1;

}

public bool Overflow()

{

if(this.top>=array.Length)

{

return true;

}

return false;

}

public int Peek()

{

if (!UnderFlow())

{

return array[--this.top];

}

else

return -1;

}

public bool UnderFlow()

{

if (top >0)

{

return false;

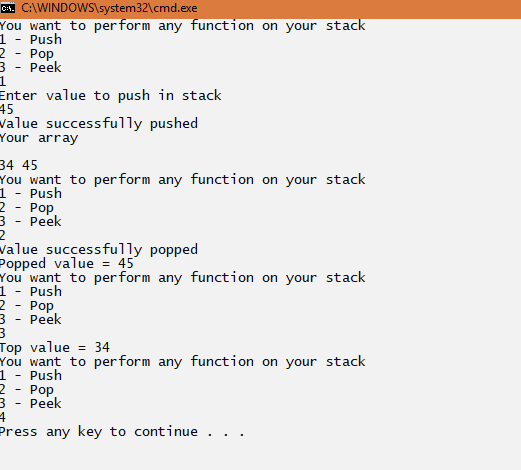
}

return true;

}

}

**Output:**



**QUESTION 2:** **Using Stack class write a code for max and min.**

**Code:**

**Program Class:**

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Enter the length of your stack");

int Length = Convert.ToInt32(Console.ReadLine());

Stack obj = new Stack(Length);

while (true)

{

Console.WriteLine("Press 1 to add value in stack");

int Value= Convert.ToInt32(Console.ReadLine());

if (Value == 1)

{

Console.WriteLine("Enter value");

int val = Convert.ToInt32(Console.ReadLine());

if (obj.Push(val))

{

Console.WriteLine("Value successfully pushed");

}

else

{

Console.WriteLine("Stack overflow");

break;

}

}

else

{

break;

}

}

obj.Display();

int minimumvalue = obj.GetMinValue();

if (minimumvalue >= 0)

{

Console.WriteLine("Minimum Value in Stack = " + minimumvalue);

}

int maximumvalue = obj.GetMaxValue();

if (maximumvalue >= 0)

{

Console.WriteLine("Maximum Value in Stack = " + maximumvalue);

}

}

}

**Stack Class:**

class Stack

{

private int Length;

private int top=0;

private int[] array;

public Stack(int length)

{

Length = length;

array = new int[length];

}

public void Display()

{

Console.WriteLine("Your array");

Console.WriteLine();

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

{

Console.Write(array[i]+" ");

}

Console.WriteLine();

}

public bool Push(int val)

{

if (!Overflow())

{

array[Count(this.top)]=val;

return true;

}

return false;

}

public int Count(int top)

{

return this.top++;

}

public bool Overflow()

{

if(this.top>=array.Length)

{

return true;

}

return false;

}

public int GetMaxValue()

{

if (!UnderFlow())

{

int max = array[0];

for (int i = 1; i < this.top; i++)

{

if (max < array[i])

{

max = array[i];

}

}

return max;

}

else

{

return -1;

}

}

public int GetMinValue()

{

if (!UnderFlow())

{

int min = array[0];

for (int i = 1; i < this.top; i++)

{

if (min > array[i])

min = array[i];

}

return min;

}

else

{

return -1;

}

}

public bool UnderFlow()

{

if (top >0)

{

return false;

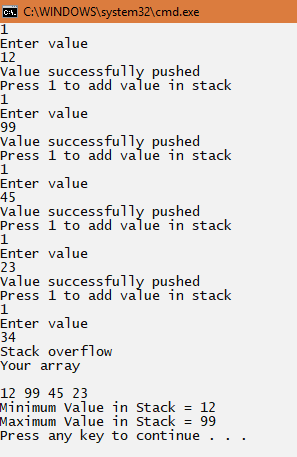
}

return true;

}

}

**Output:**



**QUESTION 3: Make Expression checker(Paranthesis) on Winform**

**Code:**

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void CheckBtn\_Click(object sender, EventArgs e)

{

string Input = InputTxt.Text;

if (ValidExpression(Input))

{

CheckLabel.Text = "Valid";

CheckLabel.ForeColor = Color.Green;

}

else

{

CheckLabel.Text = "Invalid";

CheckLabel.ForeColor = Color.Red;

}

}

private bool ValidExpression(string value)

{

char[] array = value.ToCharArray();

int left = 0, right = 0;

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

{

if (array[i] == '(') {

left++;

}

if (array[i] == ')')

{

right++;

}

}

if (left == right)

{

return true;

}

return false;

}

private void btnclose\_Click(object sender, EventArgs e)

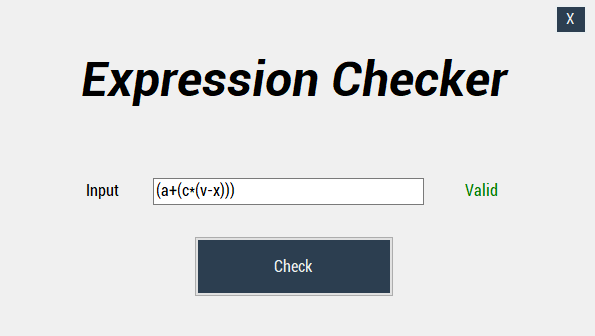
{

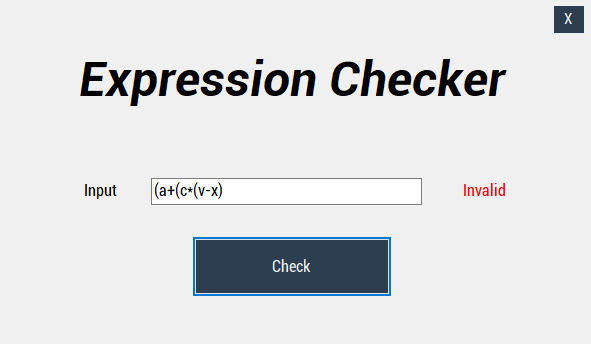
Application.Exit();

}

}

**Output:**





**QUESTION 4:Take a String as input and print its reverse using your Stack class**

**Code:**

**Program Class:**

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Enter a string");

string Value = Console.ReadLine();

Stack obj = new Stack(Value);

string ReverseValue = obj.ReverseString();

Console.WriteLine("Its reverse is ");

Console.WriteLine(ReverseValue);

}

}

**Stack Class:**

class Stack

{

private int Length;

private int top;

private char[] array;

public Stack(string value)

{

array = value.ToCharArray();

Length = array.Length;

top = Length - 1;

}

public string ReverseString()

{

try

{

char[] copyarray = new char[array.Length];

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

{

copyarray[i] = array[i];

}

int count = 0;

for (int i = top; i >= 0 && count<array.Length; i--,count++)

{

array[count] = copyarray[i];

}

string reverseString = "";

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

{

reverseString += array[i];

}

return reverseString;

}

catch (Exception)

{

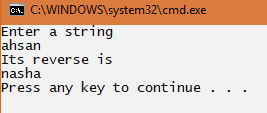
return null;

}

}

}

**Output:**



**QUESTION 5: Design & implement all methods of Simple Queue.**

**Code:**

**Program Class:**

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Enter length of Queue");

int length = Convert.ToInt32(Console.ReadLine());

Queue obj = new Queue(length);

while (true)

{

Console.WriteLine("You want to perform any function on Queue? Press 0 to exit");

Console.WriteLine("1- Enqueue\n2- Dequeue\n3- Front");

int answer = Convert.ToInt32(Console.ReadLine());

if (answer == 1)

{

Console.WriteLine("Enter value");

int val = Convert.ToInt32(Console.ReadLine());

if (obj.Enqueue(val))

{

Console.WriteLine("Value successfully enqueued");

obj.Display();

}

else

{

Console.WriteLine("Stack overflow");

}

}

else if (answer == 2)

{

int val = obj.Dequeue();

if (val >= 0)

{

Console.WriteLine("Value successfully dequeued");

Console.WriteLine("Dequeued value = " + val);

}

else

{

Console.WriteLine("Stack Underflow");

}

}

else if (answer == 3)

{

int val = obj.Front();

if (val >= 0)

{

Console.WriteLine("Front value = " + val);

}

else

{

Console.WriteLine("Stack empty");

}

}

else

{

break;

}

Console.WriteLine();

}

}

}

**Queue Class:**

class Queue

{

private int[] array;

private int Length, front, rear;

public Queue(int length) {

array = new int[length];

this.Length = length;

rear = 0;

}

public bool Enqueue(int val)

{

if (!Overflow())

{

array[rear++]=val;

return true;

}

return false;

}

public int Dequeue()

{

if (!UnderFlow())

{

return array[front++];

}

return -1;

}

public int Front()

{

if (!UnderFlow())

{

return array[front];

}

return -1;

}

public void Display()

{

Console.WriteLine("Your array");

Console.WriteLine();

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

{

Console.Write(array[i] + " ");

}

Console.WriteLine();

}

public bool UnderFlow()

{

if (rear > 0)

{

return false;

}

return true;

}

public bool Overflow()

{

if (this.rear >= array.Length)

{

return true;

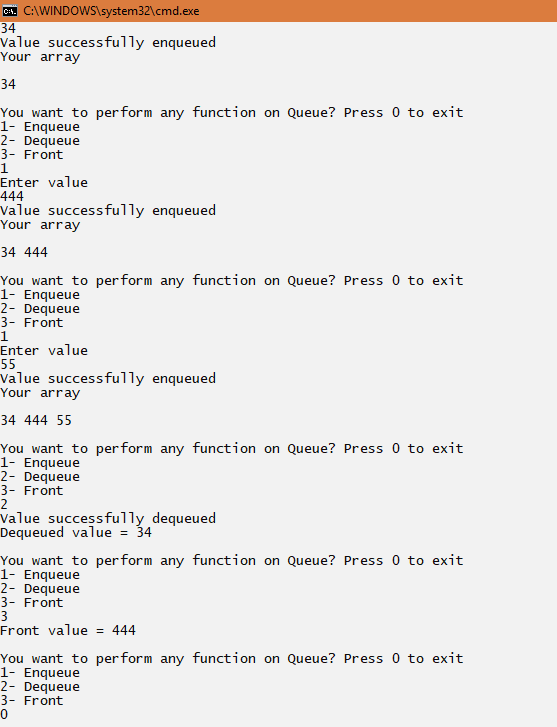
}

return false;

}

}

**Output:**



**QUESTION 6:Design & implement all methods of Circular Queue.**

**Code:**

**Program Class:**

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Enter length of Queue");

int length = Convert.ToInt32(Console.ReadLine());

Queue obj = new Queue(length);

while (true)

{

Console.WriteLine("You want to perform any function on Queue? Press 0 to exit");

Console.WriteLine("1- Enqueue\n2- Dequeue\n3- Front");

int answer = Convert.ToInt32(Console.ReadLine());

if (answer == 1)

{

Console.WriteLine("Enter value");

int val = Convert.ToInt32(Console.ReadLine());

if (obj.Enqueue(val))

{

Console.WriteLine("Value successfully enqueued");

obj.Display();

}

else

{

Console.WriteLine("Stack overflow");

}

}

else if (answer == 2)

{

int val = obj.Dequeue();

if (val >= 0)

{

Console.WriteLine("Value successfully dequeued");

Console.WriteLine("Dequeued value = " + val);

}

else

{

Console.WriteLine("Queue Underflow");

}

}

else if (answer == 3)

{

int val = obj.Front();

if (val >= 0)

{

Console.WriteLine("Front value = " + val);

}

else

{

Console.WriteLine("Queue empty");

}

}

else

{

break;

}

Console.WriteLine();

}

}

}

**Queue Class**

class Queue

{

private int[] array;

private int Length, front, count=0, rear;

public Queue(int length) {

array = new int[length];

this.Length = length;

rear = -1;

front = 0;

}

public int RearCount()

{

if(rear == -1)

{

return ++rear;

}

rear = (rear + 1) % Length;

return rear;

}

public int FrontCount()

{

front = (front + 1) % Length;

return front;

}

public bool Enqueue(int val)

{

if (!Overflow())

{

rear = RearCount();

array[rear]=val;

count++;

return true;

}

return false;

}

public int Dequeue()

{

if (!UnderFlow())

{

int val = array[front];

front = FrontCount();

count--;

return val;

}

return -1;

}

public int Front()

{

if (!UnderFlow())

{

return array[front];

}

return -1;

}

public void Display()

{

Console.WriteLine("Your array");

Console.WriteLine();

for (int i = front-1; i <rear && i<Length-1; i = (i + 1)%Length)

{

Console.Write(array[i+1] + " ");

}

Console.WriteLine();

}

public bool UnderFlow()

{

if (count <= 0)

return true;

return false;

}

public bool Overflow()

{

if (count >= Length)

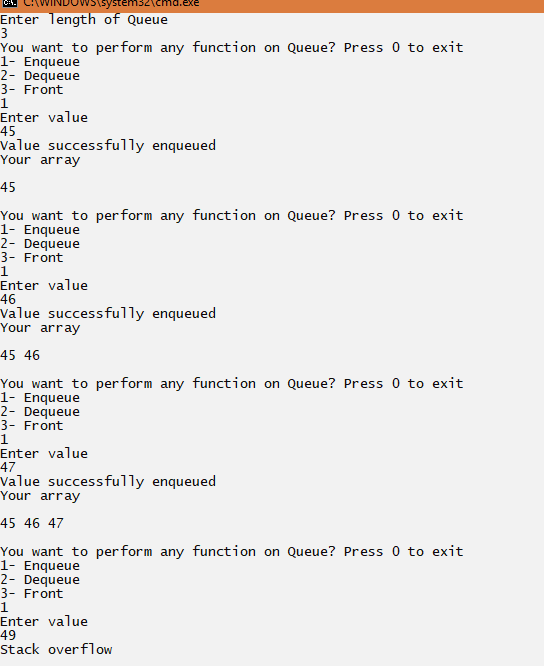
return true;

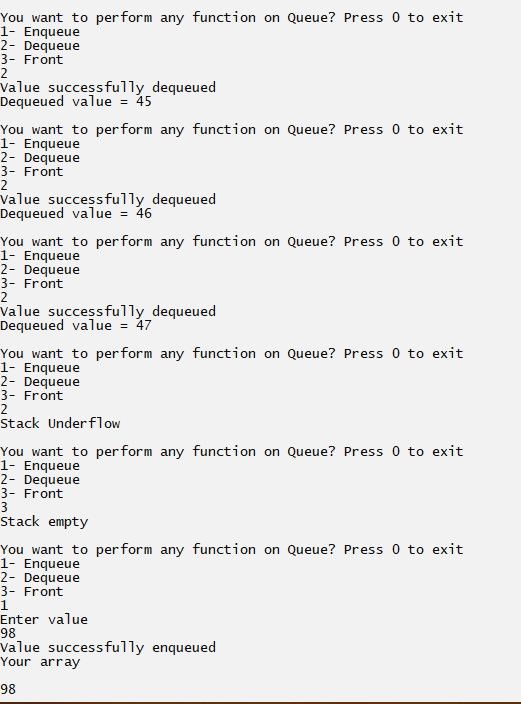
return false;

}

}

**Output:**





**QUESTION 7:Design and implement for Priority Queue.**

**Code:**

**PriortyQueue Class:**

class PriortyQueue

{

int[,] array;

int Length, front, rear;

public PriortyQueue(int length)

{

Length =length;

array = new int[length, 2];

front = 0;

rear = 0;

}

public bool SortHigh(int value, int x)

{

int temp = rear-1;

if (!UnderFlow())

{

for (int i = rear-1; i >=0; i--)

{

if(array[i,0]==2 || array[i, 0] == 3) {

array[i + 1, 0] = array[i, 0];

array[i + 1, 1] = array[i, 1];

}

else

{

temp = i;

break;

}

}

//while ((array[temp, 0] == 2 || array[temp, 0] == 3) && temp >= 0)

//{

// if (temp <= rear && temp > 0)

// {

// array[temp+1, 0] = array[temp , 0];

// array[temp+1, 1] = array[temp , 1];

// temp--;

// }

// else

// {

// break;

// }

//}

if (Enqueue(value, x, temp))

{

return true;

}

else return false;

}

else

return Enqueue(value, x);

}

public int Dequeue()

{

if (!UnderFlow())

{

int val = array[front, 1];

for (int i = front; i < rear-1; i++)

{

array[i, 0] = array[i + 1, 0];

array[i, 1] = array[i + 1, 1];

}

rear--;

return val;

}

return -1;

}

public int Front()

{

if (!UnderFlow())

{

return array[front, 1];

}

return -1;

}

public bool Enqueue(int val, int j)

{

if (!Overflow())

{

array[rear, 1] = val;

array[rear, 0] = j;

rear++;

return true;

}

return false;

}

public bool Enqueue(int val, int j,int indx)

{

if (!Overflow())

{

if (indx >= 0)

{

array[indx, 1] = val;

array[indx, 0] = j;

rear++;

return true;

}

else

{

array[0, 1] = val;

array[0, 0] = j;

rear++;

return true;

}

}

return false;

}

public bool Overflow()

{

if (rear<Length)

return false;

return true;

}

public bool SortMedium(int value, int x)

{

int temp = rear;

if (!UnderFlow())

{

while (array[temp, 0] == 3 && temp >= 0)

{

if (temp < 0)

{

break;

}

else

{

if (temp < Length - 1)

{

array[temp + 1, 0] = array[temp, 0];

array[temp + 1, 1] = array[temp, 1];

temp--;

}

else

{

break;

}

}

}

if (temp >= 0 && temp < Length)

{

if (Enqueue(value, x, temp ))

{

return true;

}

else

{

return false;

}

}

}

else

return Enqueue(value, x);

return false;

}

public bool UnderFlow()

{

if (rear==front)

return true;

return false;

}

public void Display()

{

Console.WriteLine("Your array");

Console.WriteLine("Value\tPriorty");

for (int i = front; i<rear;i++)

{

Console.WriteLine(array[i , 1] + "\t" + array[i, 0]);

}

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

}

**Program Class:**

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Enter length of Queue");

int length = Convert.ToInt32(Console.ReadLine());

PriortyQueue obj = new PriortyQueue(length);

while (true)

{

Console.WriteLine("You want to perform any function on Queue? Press 0 to exit");

Console.WriteLine("1- Enqueue\n2- Dequeue\n3- Front");

int answer = Convert.ToInt32(Console.ReadLine());

if (answer == 1)

{

int priorty, tempvalue;

Console.WriteLine("Enter priorty");

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

Console.WriteLine("Enter value");

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

if (priorty == 1)

{

if (obj.SortHigh(tempvalue, priorty))

{

Console.WriteLine("Value Enqueued");

obj.Display();

}

else

{

Console.WriteLine("Queue Overflow");

}

}

else if (priorty == 2)

{

if (obj.SortMedium(tempvalue, priorty))

{

Console.WriteLine("Value Enqueued");

obj.Display();

}

else

{

Console.WriteLine("Queue Overflow");

}

}

else if (priorty == 3)

{

if (obj.Enqueue(tempvalue, priorty))

{

Console.WriteLine("Value Enqueued");

obj.Display();

}

else

{

Console.WriteLine("Queue Overflow");

}

}

else

{

Console.WriteLine("Invalid priorty");

}

}

else if (answer == 2)

{

int val = obj.Dequeue();

if (val >= 0)

{

Console.WriteLine("Value successfully dequeued");

Console.WriteLine("Dequeued value = " + val);

}

else

{

Console.WriteLine("Stack Underflow");

}

}

else if (answer == 3)

{

int val = obj.Front();

if (val >= 0)

{

Console.WriteLine("Front value = " + val);

}

else

{

Console.WriteLine("Stack empty");

}

}

else

{

break;

}

}

}

}

**Output:**

