 **BAHRIA UNIVERSITY (KARACHI CAMPUS)**

**ASSGINMENT # 2 - FALL 2019**

# Data Structures and Algorithm (CSC-221)

Class: **BSE 3 B** Submission Deadline (BSE 3 B): 26 **December, 2019**

Student Name: **SYED ALI ABBAS**

Lab Instructor: **Engr. Saniya Sarim** Max Marks: 10

**QUESTION-1:** Using dynamic Stack class write a code which takes n number of inputs from user and create sorted list. I.e. in ascending order.

**CODE:**

**Class Stack:**

class Stack

{

static int max = 3000;

int top;

int[] stack = new int[max];

public Stack()

{

top = -1;

}

public bool underflow()

{

if (top < 0)

{

Console.WriteLine("UnderFlow");

return true;

}

else

{

return false;

}

}

public bool Overflow()

{

if (top >= max)

{

Console.WriteLine("Overflow");

return true;

}

else

{

return false;

}

}

public bool push(int value)

{

if (Overflow())

{

return false;

}

else

{

stack[++top] = value;

return true;

}

}

public int pop()

{

if (underflow())

{

return 0;

}

else

{

int val = stack[top--];

return val;

}

}

public int peek()

{

if (underflow())

{

return 0;

}

else

{

return stack[top];

}

}

public void display()

{

if (underflow())

{

Console.WriteLine(" Stack Empty");

}

else

{

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

{

Console.WriteLine("Stack index {0} = {1}", i, stack[i]);

}

}

}

}

}

**Main:**

static void Main(string[] args)

{

Console.WriteLine("Enter Length of Stack : ");

int Alength = int.Parse(Console.ReadLine());

int[] arr = new int[Alength];

Console.WriteLine("Enter Elements : ");

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

{

arr[i] = int.Parse(Console.ReadLine());

}

Console.WriteLine("Before Sorting : ");

foreach (int item1 in arr)

{

Console.Write(item1 + " ");

}

Console.WriteLine();

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

{

for (int j = 1; j < arr.Length; j++)

{

int k = j - 1;

if (arr[k] > arr[j])

{

int temp = arr[k];

arr[k] = arr[j];

arr[j] = temp;

}

}

}

Console.WriteLine("After Sorting : ");

foreach (int item in arr)

{

Console.Write(item + " ");

}

Stack st = new Stack();

do

{

Console.WriteLine();

Console.WriteLine("Select Option \n1 = PUSH array Values To Stack \n2 = POP \n3 = View Peek \n0 = Exit");

int slct = int.Parse(Console.ReadLine());

if (slct == 1)

{

Console.WriteLine();

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

{

st.push(arr[i]);

}

Console.WriteLine("Stack Values : ");

st.display();

}

else if (slct == 2)

{

Console.Write("" + st.pop() + " is poped\n");

Console.WriteLine("Remaning Stack elements Are");

st.display();

}

else if (slct == 3)

{

Console.WriteLine("" + st.peek()+ "is at Peek");

}

else if (slct == 0)

{

break;

}

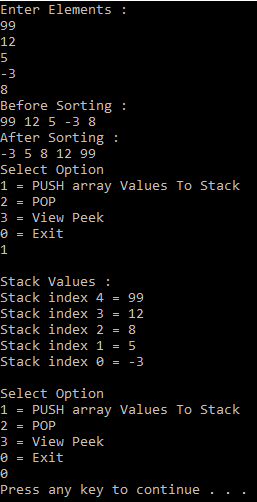
} while (true);

}

}

}

**OUTPUT:**



**QUESTION-2:** Using dynamic queue class write a code which takes n number of inputs from user and create sorted list in descending order.

**CODE:**

**Class Queue:**

class DQueue

{

int[] arr;

int front;

int max;

int rear;

public DQueue(int size)

{

arr = new int[size];

front = 0;

rear = -1;

max = size;

}

public void push(int item)

{

if (rear == max - 1)

{

Console.WriteLine("OVER-FLOW");

return;

}

else

{

arr[++rear] = item;

}

}

public int pop()

{

if (front == rear + 1)

{

Console.WriteLine("UNDER FLOW");

return -1;

}

else

{

Console.WriteLine("deleted element is: " + arr[front]);

return arr[front++];

}

}

public int peek()

{

if (front < 0)

{

return 0;

}

else

{

return arr[rear];

}

}

public void display()

{

if (front == rear + 1)

{

Console.WriteLine("Queue is Empty");

return;

}

else

{

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

{

Console.WriteLine("Queue" + (i + 1) + ": " + arr[i]);

}

}

}

}

}

**Main:**

static void Main(string[] args)

{

Console.WriteLine("ENTER QUEUE LENGTH : ");

int Alength = int.Parse(Console.ReadLine());

int[] arr = new int[Alength];

Console.WriteLine("Enter Element : ");

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

{

arr[i] = int.Parse(Console.ReadLine());

}

Console.WriteLine("Array Before Sorting");

foreach (int item1 in arr)

{

Console.Write(item1 + " ");

}

Console.WriteLine();

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

{

for (int j = 1; j < arr.Length; j++)

{

int k = j - 1;

if (arr[k] > arr[j])

{

int temp = arr[k];

arr[k] = arr[j];

arr[j] = temp;

}

}

}

Console.WriteLine("AFTER SORTING");

for (int i = arr.Length - 1; i >= 0; i--)

{

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

}

DQueue LQ = new DQueue(arr.Length);

do

{

Console.WriteLine();

Console.WriteLine("Select Option \n1 = ENQUEUE \n2 =DEQUEUE \n3 = FRONT \n0 = Exit");

int slct = int.Parse(Console.ReadLine());

if (slct == 1)

{

Console.WriteLine();

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

{

LQ.push(arr[i]);

}

Console.WriteLine("Values in QUEUE Are: ");

LQ.display();

}

else if (slct == 2)

{

Console.Write("DEQUEUED = " + LQ.pop() + "\n");

Console.WriteLine(" Queue elements Left Are");

LQ.display();

}

else if (slct == 3)

{

Console.WriteLine("FRONT Value is = " + LQ.peek());

}

else if (slct == 0)

{

break;

}

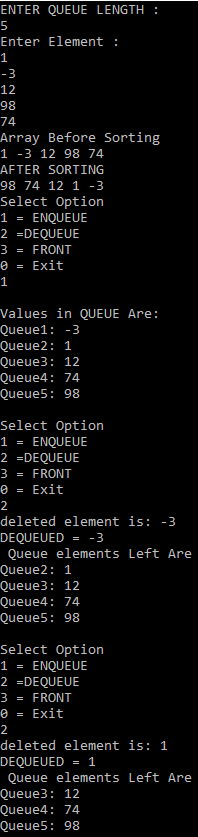
} while (true);

}

}

}

**OUTPUT:**



**QUESTION-3:** With the help of Stacks, implement Polish notation in which you have to convert given expression to postfix notation. Show all the conversion using arrays or linked list.

**CODE:**

**Class Nodes:**

Class nodes:

{

internal Nodes next;

internal char data;

public Nodes()

{

next = null;

}

public Nodes(char val)

{

this.data = val;

}

public Nodes(Nodes obj)

{

this.next = obj;

}

}

**Linked Lists:**

class Stacks

{

Nodes start;

public Stacks()

{

start = new Nodes();

}

public Stacks(Nodes obj)

{

start = obj;

}

public bool Underflow()

{

if (start.next != null)

{

return false;

}

else

{

return true;

}

}

public bool Push(char val)

{

Nodes n = new Nodes(val);

if (!Underflow())

{

n.next = start.next;

start.next = n;

return true;

}

else

{

start.next = n;

return true;

}

}

public char Peek()

{

if (!Underflow())

{

char val;

val = (char)start.next.data;

return val;

}

return ' ';

}

public char Pop()

{

if (!Underflow())

{

char val;

if (start.next.next != null)

{

val = (char)start.next.data;

start.next = start.next.next;

return val;

}

else

{

val = (char)start.next.data;

start.next = null;

return val;

}

}

return ' ';

}

public void Display()

{

if (!Underflow())

{

Nodes temp = start.next;

Console.WriteLine("Your list");

while (temp.next != null)

{

Console.Write(temp.data + " ");

temp = temp.next;

}

Console.Write(temp.data);

Console.WriteLine();

}

else

{

Console.WriteLine("List empty");

}

}

}

}

**Main:**

static void Main(string[] args)

{

Console.WriteLine("Enter Exp to convert into PostFix : ");

string value = Console.ReadLine();

if (ValidExpression(value))

{

string postfix = IntoPost(value);

Console.WriteLine("Postfix = " + postfix);

}

}

public static string IntoPost(string infix)

{

Stacks obj = new Stacks();

char[] arr = infix.ToCharArray();

char[] postarr = new char[arr.Length];

int count = 0;

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

{

int ascii = (int)arr[i];

if (ascii >= 48 && ascii <= 57)

{

postarr[count] = arr[i];

count++;

}

else if ((ascii >= 65 && ascii <= 90) || (ascii >= 97 && ascii <= 122))

{

postarr[count] = arr[i];

count++;

}

else if (arr[i] == '(' || arr[i] == '^')

{

obj.Push(arr[i]);

}

else if (arr[i] == '\*' || arr[i] == '/' || arr[i] == '%')

{

char temp = obj.Peek();

if (temp == '(' || temp == '-' || temp == '+')

{

obj.Push(arr[i]);

}

else if (temp == '\*' || temp == '/' || temp == '%')

{

postarr[count] = obj.Pop();

count++;

obj.Push(arr[i]);

}

else if (temp == '^')

{

postarr[count] = obj.Pop();

count++;

char temp1 = obj.Peek();

if (temp1 == '\*' || temp1 == '/' || temp1 == '%')

{

postarr[count] = obj.Pop();

count++;

obj.Push(arr[i]);

}

else

{

obj.Push(arr[i]);

}

}

else

{

obj.Push(arr[i]);

}

}

else if (arr[i] == '+' || arr[i] == '-')

{

char temp = obj.Peek();

if (temp == '^')

{

postarr[count] = obj.Pop();

count++;

char temp1 = obj.Peek();

if (temp1 == '\*' || temp1 == '/' || temp1 == '%')

{

postarr[count] = obj.Pop();

count++;

}

char temp2 = obj.Peek();

if (temp2 == '+' || temp2 == '-')

{

postarr[count] = obj.Pop();

count++;

obj.Push(arr[i]);

}

else

{

obj.Push(arr[i]);

}

}

else if (temp == '\*' || temp == '/' || temp == '%')

{

postarr[count] = obj.Pop();

count++;

char temp1 = obj.Peek();

if (temp1 == '+' || temp1 == '-')

{

postarr[count] = obj.Pop();

count++;

obj.Push(arr[i]);

}

else

{

obj.Push(arr[i]);

}

}

else if (temp == '+' || temp == '-')

{

postarr[count] = obj.Pop();

count++;

obj.Push(arr[i]);

}

else

{

obj.Push(arr[i]);

}

}

else if (arr[i] == ')')

{

char temp = obj.Peek();

while (temp != '(')

{

postarr[count] = obj.Pop();

count++;

temp = obj.Peek();

}

obj.Pop();

}

else

{

}

}

while (!obj.Underflow())

{

postarr[count] = obj.Pop();

count++;

}

string post = "";

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

{

post += postarr[i];

}

return post;

}

private static 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;

}

}

**OUTPUT:**

