Name : Dev Adnani
SID : 202212012
Subject : DSA
Topic : Stack-Queue
Lab : 04

Q1 :

```
#include <iostream>
using namespace std;

class Stack
{
        int top;
        int capacity;
        int *arr;

public:
        Stack(int data)
        {
        top = -1;
        capacity = data;
        arr = new int[capacity];
        }
        void push(int data)
        {
        if (isFull())
        {
        return;
        }
        else
        {
        top++;
        arr[top] = data;
        }
        }
        int pop()
        {
        if (isEmpty())
        {
        cout << "Stack Is Empty" << endl;
        return -1;
        }
        else
        {
        top--;
        int val = arr[top];
        return val;
        }
        }
```

```
        int peek()
        {
        if (isEmpty())
        {
        cout << "Stack Is Empty" << endl;
        return -1;
        }
        return arr[top];
        }
        int size()
        {
        return capacity;
        }
        bool isEmpty()
        {
        if (top == -1)
        return true;
        else
        return false;
        }
        bool isFull()
        {
        if (top == capacity - 1)
        return true;
        else
        return false;
        }
};

int main()
{
        cout << " Enter Stack Size  :";
        int n;
        cin >> n;

        Stack st(n);

        int check;
        while (n != 7)
        {
        cout << "Enter Choice :" << endl;
        cout << "Enter 1 For Push :" << endl;
        cout << "Enter 2 For Pop :" << endl;
        cout << "Enter 3 For Peeking:" << endl;
```

```
cout << "Enter 4 For Size:" << endl;
cout << "Enter 5 For Checking If Stack Is Full:" << endl;
cout << "Enter 6 For Checking If Stack Is Empty:" << endl;
cout << "Enter 7 For Exit:" << endl;

cin >> check;
switch (check)
{
case 1:
{
cout << "Enter Element:";
int k;
cin >> k;
st.push(k);
break;
}
case 2:
{
cout << "Poped Data : " << endl;
break;
}
case 3:
{
cout << "Peeking Data : " << st.peek() << endl;
break;
}
case 4:
{
cout << "Size Of Stack : " << st.size() << endl;
break;
}
case 5:
{
cout << "Full  : " << st.isFull() << endl;
break;
}
case 6:
{
cout << "Empty  : " << st.isEmpty() << endl;
break;
}
case 7:
exit;
break;
```

```
        default:
        break;
        }
        }
}
```

O/P :

```
PS F:\D> cd "f:\D"
PS F:\D> cd "f:\D\" ; if ($?) { g++ d.cpp -o d } ; if ($?) { .\d }
 Enter Stack Size  :3
Enter Choice :
Enter 1 For Push :
Enter 2 For Pop :
Enter 3 For Peeking:
Enter 4 For Size:
Enter 5 For Checking If Stack Is Full:
Enter 6 For Checking If Stack Is Empty:
Enter 7 For Exit:
1
Enter Element:20
Enter Choice :
Enter 1 For Push :
Enter 2 For Pop :
Enter 3 For Peeking:
Enter 4 For Size:
Enter 5 For Checking If Stack Is Full:
Enter 6 For Checking If Stack Is Empty:
Enter 7 For Exit:
3
Peeking Data : 20
Enter Choice :
Enter 1 For Push :
Enter 2 For Pop :
Enter 3 For Peeking:
Enter 4 For Size:
Enter 5 For Checking If Stack Is Full:
Enter 6 For Checking If Stack Is Empty:
Enter 7 For Exit:
4
Size Of Stack : 3
```

Q2 : Given string str, we need to print reverse of individual words

```
#include <iostream>
#include <string.h>
using namespace std;
#define mx 100

class Stack
{
	char arr[mx];
	int top = -1;

public:
	void ph(char ch)
	{
	if (top >= mx - 1)
	{
	cout << "Stack Overflow" << endl;
	}
	else if (top < 0)
	{
	top = 0;
	arr[top] = ch;
	}
	else
	{
	top++;
	arr[top] = ch;
	}
	}

	char pop()
	{
	if (top < 0)
	{
	return '1';
	}
	else if (top == mx - 1)
	{
	return arr[top];
	top = -1;
	}
	else
	{
```

```
        char temp = arr[top];
        top--;
        return temp;
        }
        }
};

void RevStr(string x)
{
        Stack s1;
        int i = 0, size;
        size = x.length();
        while (i <= size)
        {
        if (x[i] != ' ' && i < size)
        {
        s1.ph(x[i]);
        i++;
        }
        else
        {
        char ch = s1.pop();
        if (ch == '1')
        {
                i++;
                cout << " ";
        }
        else
        {
                cout << ch;
        }
        }
        }
}

int main()
{
        char d[100];
        cout << "Enter a string: ";
        cin.get(d, 100);
        RevStr(d);
        return 0;
}
```

O/P :


Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS F:\D> cd "f:\D"
PS F:\D> cd "f:\D\" ; if ($?) { g++ d.cpp -o d } ; if ($?) { .\d }
Enter a string: Dev Adnani
veD inandA
PS F:\D>

Q3 : Given an array, print the Next Greater Element (NGE) for every element. The Next greater Element for an element x is the first greater element on the right side of x in the array. Elements for which no greater element exists, consider the next greater element as -1

```
#include <iostream>
#include <string.h>
using namespace std;

void nextGrElement(int arrx[], int a)
{
	int next, i, j;
	for (i = 0; i < a; i++)
	{
	next = -1;
	for (j = i + 1; j < a; j++)
	{
	if (arrx[i] < arrx[j])
	{
		next = arrx[j];
		break;
	}
	}
	cout << arrx[i] << " -> " << next << endl;
	}
}
int main()
{
	int n;
	cout << "Enter Arr Size : ";
	cin >> n;
	int arrx[n];
	cout << "Enter Elements: ";
	for (int i = 0; i < n; i++)
	cin >> arrx[i];
	nextGrElement(arrx, n);
	return 0;
}
```

O/P :

```
PS F:\D> cd "f:\D"
PS F:\D> cd "f:\D\" ; if ($?) { g++ d.cpp -o d } ; if ($?) { .\d }
Enter Arr Size : 5
Enter Elements: 10 20 30 10 20
10 -> 20
20 -> 30
30 -> -1
10 -> 20
20 -> -1
PS F:\D>
```

Q4 : Write a program to convert infix expression to postfix expression

```
#include <iostream>
#include <string.h>
using namespace std;

class Stack
{
        int top;
        int MAX;
        int *a;

public:
        Stack(int size)
        {
        top = -1;
        MAX = size;
        a = new int[MAX];
        }
        bool isEmpty()
        {
        return (top < 0);
        }

        bool isFull()
        {
        return (top == MAX - 1);
        }

        int peek()
        {
        return a[top];
        }
        bool push(int x)
        {
        if (top >= (MAX - 1))
        {
        cout << "Stack Overflow";
        return false;
        }
        else
        {
        top++;
        a[top] = x;
```

```
        return true;
        }
        }
        int pop()
        {
        if (top < 0)
        {
        cout << "Stack Underflow";
        return -1;
        }
        else
        {
        int x = a[top];
        top--;
        return x;
        }
        }
};

int precedence(char x)
{
        if (x == '+' || x == '-')
        return 1;
        if (x == '*' || x == '/')
        return 2;
        if (x == '^')
        return 3;
        return 0;
}

string cvt(string str)
{
        int i = 0;
        string posfix = "";

        Stack s(100);
        while (str[i] != '\0')
        {
        if (str[i] >= 'a' && str[i] <= 'z' || str[i] >= 'A' && str[i] <= 'Z')
        {
        posfix += str[i];
        i++;
        }
        else if (str[i] == '('
```

```
        {
        s.push(str[i]);
        i++;
        }

        else if (str[i] == ')')
        {
        while (s.peek() != '(')
                posfix += s.pop();
        s.pop();
        i++;
        }
        else
        {
        while (!s.isEmpty() && precedence(str[i]) <= precedence(s.peek()))
        {
                posfix += s.pop();
        }
        s.push(str[i]);
        i++;
        }
        }
        while (!s.isEmpty())
        {
        posfix += s.pop();
        }
        cout << "postfix is : " << posfix;
        return posfix;
}

int main()
{
        string str;
        cout << "Enter your string : ";
        cin >> str;
        string posfix;

        posfix = cvt(str);
        return 0;
}
```

O/P:

```
PS F:\D> cd "f:\D"
PS F:\D> cd "f:\D\" ; if ($?) { g++ d.cpp -o d } ; if ($?) { .\d }
Enter your string : a+b+d
postfix is : ab+d+
PS F:\D>
```

Q5 : Write a program to implement two stacks with a single array. Create separate push() and pop() for both the stacks and perform a set of push and pop such that overflow, underflow, successful push, as well as successful pop operation, takes place for both the stacks.

Note: Utilize the entire space of the array

```
#include <iostream>
#include <string.h>
using namespace std;

class Stack
{
        int *arr;
        int top1 = -1, top2;
        int capacity;

public:
        Stack(int n)
        {
        this->capacity = n;
        this->top2 = n;
        arr = new int[capacity];
        }
        void phFirst(int d)
        {
        if (top1 == capacity - 1 || top1 == top2 - 1)
        {
        cout << "Stack1 Overflow";
        return;
        }

        arr[++top1] = d;
        return;
        }
        void phSecond(int d)
        {

        if (top2 == 0 || top2 == top1 + 1)
        {
        cout << "Stack2 Overflow";
        return;
        }
        arr[--top2] = d;
        }
```

```
        int popFirstSt()
        {
        if (top1 == -1)
        {
        cout << "Stack1 UnderFlow";
        return -1;
        }
        else
        return arr[top1--];
        }
        int popSecondSt()
        {
        if (top2 == capacity)
        {
        cout << "Stack2 UnderFlow";
        return -1;
        }
        else
        return arr[top2++];
        }
};

int main()
{
        int n, d;
        cout << "Enter Size : ";
        cin >> n;
        Stack s1(n);
        while (d != 5)
        {
        cout << endl;
        cout << "1 : Push In First Stack" << endl;
        cout << "2 : Push In Second Stack" << endl;
        cout << "3 : Pop From First Stack" << endl;
        cout << "4 : Pop From Second Stack" << endl;
        cout << "5 : Exit" << endl;
        cout << "Enter your choice : ";
        cin >> d;
        switch (d)
        {
        case 1:
        {
        int d;
        cout << "Enter : ";
```

```
        cin >> d;
        s1.phFirst(d);
        }
        break;
        case 2:
        {
        int d;
        cout << "Enter : ";
        cin >> d;
        s1.phSecond(d);
        }
        break;
        case 3:
        {
        s1.popFirstSt();
        cout << "Popped From Stck 1" << endl;
        }
        break;
        case 4:
        {
        s1.popSecondSt();
        cout << "Popped From Stck 2 :" << endl;
        }
        }
        }
        return 0;
}
```

O/P:

```
PS F:\D> cd "f:\D\" ; if ($?) { g++ d.cpp -o d } ; if ($?) { .\d }
Enter Size : 4

1 : Push In First Stack
2 : Push In Second Stack
3 : Pop From First Stack
4 : Pop From Second Stack
5 : Exit
Enter your choice : 1
Enter : 20

1 : Push In First Stack
2 : Push In Second Stack
3 : Pop From First Stack
4 : Pop From Second Stack
5 : Exit
Enter your choice : 1
Enter : 20

1 : Push In First Stack
2 : Push In Second Stack
3 : Pop From First Stack
4 : Pop From Second Stack
5 : Exit
Enter your choice : 3
Popped From Stck 1

1 : Push In First Stack
2 : Push In Second Stack
3 : Pop From First Stack
4 : Pop From Second Stack
5 : Exit
Enter your choice : 3
Popped From Stck 1
```

Q6 :

```
#include <iostream>
#include <string.h>
using namespace std;

class Queue
{
        int *arr;
        int capacity;
        int front = -1;
        int rear = -1;

public:
        Queue(int capacity)
        {
        this->capacity = capacity;
        arr = new int[capacity];
        }
        bool empty()
        {
        return (front == -1 && rear == -1) ? true : false;
        }
        void enQueue(int element)
        {
        if (rear == -1 && front == -1)
        {
        front++;
        arr[++rear] = element;
        return;
        }

        if (rear == capacity - 1)
        {
        cout << "Queue is Full" << endl;
        return;
        }
        else
        {
        rear = rear + 1;
        arr[rear] = element;
        return;
        }
        }
```

```
        int deQueue()
        {
        if (front == -1 && rear == -1)
        {
        cout << "Queue is empty" << endl;
        return -1;
        }
        if (rear == front)
        {
        int k = arr[front];
        front = -1;
        rear = -1;
        return k;
        }
        return arr[front++];
        }
        int checkFront()
        {
        if (front == -1 && rear == -1)
        {
        cout << "Queue is empty" << endl;
        return -1;
        }
        return arr[front];
        }
};

int main()
{
        cout << "Enter Size For Queue : ";
        int n, d = 0;
        cin >> n;
        Queue q1(n);
        while (d != 5)
        {
        cout << endl;
        cout << "1: enQueue" << endl;
        cout << "2: deQueue" << endl;
        cout << "3: Check Is Empty" << endl;
        cout << "4: Front Element" << endl;
        cout << "5: Exit" << endl;
        cout << "Enter your choice : ";
        cin >> d;
        switch (d)
```

```
{
case 1:
{
int k;
cout << "Enter your element : " << endl;
cin >> k;
q1.enQueue(k);
}
break;
case 2:
{
int k;
k = q1.deQueue();
if (k != -1)
{
        cout << "Deleted element : " << k << endl;
}
}
break;

case 3:
{
if (q1.empty())
{
        cout << "Yes , Queue is empty" << endl;
}
else
{
        cout << "No , Queue is not empty" << endl;
}
}
break;
case 4:
{
int k;
k = q1.checkFront();
if (k != -1)
{
        cout << "Front element is :" << k << endl;
}
}
break;
}
}
```

```
        return 0;
}
```

O/P:

```
PS F:\D> cd "f:\D"
PS F:\D> cd "f:\D\" ; if ($?) { g++ d.cpp -o d } ; if ($?) { .\d }
Enter Size For Queue : 4

1: enQueue
2: deQueue
3: Check Is Empty
4: Front Element
5: Exit
Enter your choice : 1
Enter your element :
2

1: enQueue
2: deQueue
3: Check Is Empty
4: Front Element
5: Exit
Enter your choice : 3
No , Queue is not empty

1: enQueue
2: deQueue
3: Check Is Empty
4: Front Element
5: Exit
Enter your choice : 4
Front element is :2
```

Q7:

```
#include <iostream>
using namespace std;

class CircularQueue
{
	int *arr;
	int capacity;
	int fnt = -1;
	int rear = -1;

public:
	CircularQueue(int capacity)
	{
	this->capacity = capacity;
	arr = new int[capacity];
	}
	bool IsEmpty()
	{
	return (fnt == -1 && rear == -1) ? true : false;
	}
	void EnQueue(int element)
	{
	if (rear == -1 && fnt == -1)
	{
	fnt++;
	arr[++rear] = element;
	return;
	}

	if ((rear + 1) % capacity == fnt)
	{
	cout << "Queue is Full\n";
	return;
	}
	else
	{
	rear = (rear + 1) % capacity;
	arr[rear] = element;
	return;
	}
	}
	int DeQueue()
```

```
        {
        if (fnt == -1 && rear == -1)
        {
        cout << "Queue is empty"<<endl;
        return -1;
        }
        if (rear == fnt)
        {
        int ele = arr[fnt];
        fnt = -1;
        rear = -1;
        return ele;
        }
        int ele = arr[fnt];

        fnt = (fnt + 1) % capacity;
        return ele;
        }
        int getfnt()
        {
        if (fnt == -1 && rear == -1)
        {
        cout << "Queue is empty"<<endl;
        return -1;
        }

        return arr[fnt];
        }
};

int main()
{
        cout << "Enter the Capacity of the Queue : ";
        int s, ch,x;
        cin >> s;
        CircularQueue q(s);
        while (ch != 5)
        {
        cout << "1 : Enqueue"<<endl;
        cout << "2 : Dequeue"<<endl;
        cout << "3 : Check IsEmpty"<<endl;
        cout << "4 : Front Element"<<endl;
        cout << "5 : Exit"<<endl;
        cout << "Enter your choice : ";
```

```
cin >> ch;
switch (ch)
{
case 1:
{
cout << "Enter your element : "<<endl;
cin >> x;
q.EnQueue(x);
}
break;
case 2:
{

x = q.DeQueue();
if (x != -1)
{
        cout << "Deleted element : " << x<<endl;
}
}
break;
case 3:
{
if (q.IsEmpty())
{
        cout << "Yes , Queue is empty"<<endl;
}
else
{
        cout << "No , Queue is not empty"<<endl;
}
}
break;
case 4:
{
x = q.getfnt();
if (x != -1)
{
        cout << "fnt element is :" << x << endl;
}
}
break;
}
}
```

```
        return 0;
}
```

O/P:

```
Enter the Capacity of the Queue : 2
1 : Enqueue
2 : Dequeue
3 : Check IsEmpty
4 : Front Element
5 : Exit
Enter your choice : 1
Enter your element :
20
1 : Enqueue
2 : Dequeue
3 : Check IsEmpty
4 : Front Element
5 : Exit
Enter your choice : 1
Enter your element :
20
1 : Enqueue
2 : Dequeue
3 : Check IsEmpty
4 : Front Element
5 : Exit
Enter your choice : 1
Enter your element :
23
Queue is Full
1 : Enqueue
2 : Dequeue
3 : Check IsEmpty
4 : Front Element
5 : Exit
Enter your choice : 2
Deleted element : 20
1 : Enqueue
2 : Dequeue
3 : Check IsEmpty
4 : Front Element
5 : Exit
Enter your choice : 1
Enter your element :
24
```

Q8 :

```
#include <iostream>
using namespace std;

class CircularQueue
{
        int arr[3];
        int fnt = -1, rear = -1;
        int capacity = 3;

public:
  void Enqueue(int ele)
{
        if (fnt < 0 && rear < 0)
        {
        fnt++;
        rear++;
        arr[rear] = ele;
        }
        else if ((rear + 1) % capacity == fnt)
        {
        cout << "Queue is full" << endl;
        }
        else
        {
        rear = (rear + 1) % capacity;
        arr[rear] = ele;
        }
}
int Dequeue()
{
        if (fnt < 0 && rear < 0)
        {
        return -1;
        }
        else if (rear == fnt)
        {
        int temp = arr[fnt];
        rear = fnt = -1;
        return temp;
        }
        else
        {
```

```
        int temp = arr[fnt];
        fnt = (fnt + 1) % capacity;
        return temp;
        }
}
int fntEle()
{
        if (fnt < 0 && rear < 0)
        {
        return -1;
        }
        else
        {
        return arr[fnt];
        }
}
bool isEmpty()
{
        if (fnt < 0 && rear < 0)
        {
        return true;
        }
        else
        {
        return false;
        }
}
bool contains(int ele)
{
        if (fnt < 0 && rear < 0)
        {
        return false;
        }
        int i = fnt % 3;
        while (i != rear)
        {
        if (arr[i] == ele)
        {
        return true;
        }
        i = (i + 1) % 3;
        }
        if (arr[i] == ele)
        {
```

```
        return true;
        }
        else
        {
        return false;
        }
}
};

void inData(int ans[], int n)
{
        int val;
        for (int i = 0; i < n; i++)
        {
        cin >> val;
        ans[i] = val;
        }
}
int main()
{
        int n;
        cout << "Enter The size : ";
        cin >> n;
        int ans[n];
        inData(ans, n);
        int pf = 0;
        CircularQueue q;
        for (int i = 0; i < n; i++)
        {
        if (q.contains(ans[i]) == false)
        {

        pf++;
        if (q.isEmpty() == false)
        {
                int temp = q.Dequeue();
                q.Enqueue(temp);
        }
        else
        {
                q.Enqueue(ans[i]);
        }
        }
        }
```

```
        cout << "Page Fault are : " << pf << endl;
        return 0;
}
```

O/P :

```
PS F:\D> cd "f:\D"
PS F:\D> cd "f:\D\" ; if ($?) { g++ d.cpp -o d } ; if ($?) { .\d }
Enter The size : 6
1
3
2
1
5
8
Page Fault are : 5
PS F:\D>
```

Q9 :

```
#include <iostream>
using namespace std;
#define MAX 100

class Queue
{
	int arr[MAX];

public:
	int front = -1, rear = -1;
	int capacity = MAX;
	void Enqueue(int ele)
	{
	if (front < 0 && rear < 0)
	{
	front++;
	rear++;
	arr[rear] = ele;
	}
	else if (rear == capacity - 1)
	{
	cout << "Queue is full" << endl;
	}
	else
	{
	rear++;
	arr[rear] = ele;
	}
	}
	int Dequeue()
	{
	if (front < 0 && rear < 0)
	{
	return -1;
	}
	else if (rear == front)
	{
	int temp = arr[front];
	rear = front = -1;
	return temp;
	}
	else
```

```
        {
        return arr[front++];
        }
        }
        int getFrontElement()
        {
        if (front < 0 && rear < 0)
        {
        return -1;
        }
        else
        {
        return arr[front];
        }
        }
        bool isEmpty()
        {
        if (front < 0 && rear < 0)
        {
        return true;
        }
        else
        {
        return false;
        }
        }
};

class Stack
{
        Queue a, b;

public:
        void ph(int ele)
        {
        b.Enqueue(ele);
        while (a.isEmpty() == false)
        {
        b.Enqueue(a.getFrontElement());
        a.Dequeue();
        }
        while (b.isEmpty() == false)
        {
        a.Enqueue(b.getFrontElement());
```

```
int ele = b.Dequeue();
}
}
int pop()
{
if (a.isEmpty() == true)
{
return -1;
}
else
{
return a.Dequeue();
}
}
int peek()
{
if (a.isEmpty() == true)
{
return -1;
}
else
{
return a.getFrontElement();
}
}
int size()
{
return a.rear + 1;
}
bool isEmpty()
{
return a.isEmpty();
}
bool isFull()
{
if (a.rear == a.capacity - 1)
{

return true;
}
else
{
return false;
}
```

```
        }
};
int main()
{
        int ch = 0;
        Stack s;
        while (ch != 7)
        {
        int ele;
        cout << "1 : Push Element" << endl;
        cout << "2 : Pop Element" << endl;
        cout << "3 : Peek Element" << endl;
        cout << "4 : Size of Stack" << endl;
        cout << "5 : Check is Empty" << endl;
        cout << "6 : Check Is Full" << endl;
        cout << "7 : Exit" << endl;
        cout << "Enter Choice : ";
        cin >> ch;
        switch (ch)
        {
        case 1:
        cout << "Enter Element : ";
        cin >> ele;
        s.ph(ele);
        break;
        case 2:
        ele = s.pop();
        if (ele == -1)
        {
                cout << "Stack Underflow" << endl;
        }
        else
        {
                cout << "Element Poped is : " << ele << endl;
        }
        break;
        case 3:
        ele = s.peek();
        if (ele == -1)
        {
                cout << "Stack Underflow" << endl;
        }
        else
        {
```

```
        cout << "Peek Element is : " << ele << endl;
}
break;
case 4:
int size;
size = s.size();
if (size == -1)
{
        cout << "Stack Underflow" << endl;
}
else
{
        cout << "Size is : " << size << endl;
}
break;
case 5:
bool empty;
empty = s.isEmpty();
if (empty == true)
{
        cout << "Stack is Empty" << endl;
}
else
{
        cout << "Stack is not Empty" << endl;
}
break;
case 6:
bool full;
full = s.isFull();
if (full == true)
{
        cout << "Stack is Full" << endl;
}
else
{
        cout << "Stack is not Full" << endl;
}
break;
case 7:
break;
default:
cout << "Wrong Choice!!";
break;
```

```
        }
        }

        return 0;
}
```

O/P:

```
PS F:\D> cd "f:\D"
PS F:\D> cd "f:\D\" ; if ($?) { g++ d.cpp -o d } ; if ($?) { .\d }
1 : Push Element
2 : Pop Element
3 : Peek Element
4 : Size of Stack
5 : Check is Empty
6 : Check Is Full
7 : Exit
Enter Choice : 1
Enter Element : 2
1 : Push Element
2 : Pop Element
3 : Peek Element
4 : Size of Stack
5 : Check is Empty
6 : Check Is Full
7 : Exit
Enter Choice : 2
Element Poped is : 2
1 : Push Element
2 : Pop Element
3 : Peek Element
4 : Size of Stack
5 : Check is Empty
6 : Check Is Full
7 : Exit
Enter Choice : 3
Stack Underflow
1 : Push Element
2 : Pop Element
3 : Peek Element
4 : Size of Stack
5 : Check is Empty
6 : Check Is Full
7 : Exit
```

Q10 :

```
#include <iostream>
using namespace std;

#define MAX 100

class Queue
{
	int arrx[MAX];
	int fnt = -1, rear = -1;
	int capacity = MAX;

public:
	void enqueue(int ele)
	{
	if (fnt < 0 && rear < 0)
	{
	fnt++;
	rear++;
	arrx[rear] = ele;
	}
	else if (rear == capacity - 1)
	{
	cout << "Queue is full" << endl;
	}
	else
	{
	rear++;
	arrx[rear] = ele;
	}
	}
	int DeQueue()
	{
	int sum = 0;
	if (fnt < 0 && rear < 0)
	{
	return -1;
	}
	while (fnt != -1)
	{
	if (fnt == rear)
	{
		if (fnt % 2 == 0)
```

```
                {
                sum += arrx[fnt];
                cout << '+' << arrx[fnt];
                }
                else
                {
                sum += -(arrx[fnt]);

                cout << '-' << arrx[fnt];
                }
                rear = fnt = -1;
                return sum;
        }
        if (fnt % 2 == 0)
        {
                sum += arrx[fnt];
                if (fnt != 0)
                {
                cout << '+' << arrx[fnt];
                }
                else
                {
                cout << arrx[fnt];
                }
                int ele = arrx[fnt];
                fnt++;
        }
        else
        {
                sum += -(arrx[fnt]);
                cout << '-' << arrx[fnt];
                int ele = arrx[fnt];
                fnt++;
        }
        }
        return sum;
        }
};

int main()
{
        int ch = 0;
        Queue q;
        while (ch != 3)
```

```
        {
        int e, a;
        cout << "1 : Enqueue" << endl;
        cout << "2 : Dequeue " << endl;
        cout << "3 : Exit" << endl;
        cout << "Enter Choice : ";
        cin >> ch;
        switch (ch)
        {
        case 1:
        cout << "Enter Element : ";
        cin >> e;
        q.enqueue(e);

        break;
        case 2:
        cout << "Ans : ";
        a = q.DeQueue();
        cout << " = " << a << endl;
        break;
        case 3:
        break;
        default:
        cout << "Wrong Choice!!";
        break;
        }
        }
        return 0;
}
```

O/P:

```
PS F:\D> cd "f:\D"
PS F:\D> cd "f:\D\" ; if ($?) { g++ d.cpp -o d } ; if ($?) { .\d }
1 : Enqueue
2 : Dequeue
3 : Exit
Enter Choice : 1
Enter Element : 30
1 : Enqueue
2 : Dequeue
3 : Exit
Enter Choice : 1
Enter Element : 20
1 : Enqueue
2 : Dequeue
3 : Exit
Enter Choice : 2
Ans : 30-20 = 10
1 : Enqueue
2 : Dequeue
3 : Exit
Enter Choice :
```