

Name : Amruta Pede

Reg. no. :2020Bit070

## Practical 2

Write c/ c++ code to implement concept of :

- 1) Stack using linkedlist
- 2) Queue using linkedlist
- 3) Doubly linkedlist
- 4) Enqueue
- 5) Dequeue

1)Stack using linkedlist:

```
// C++ program to Implement a stack
```

```
// using singly linked list
```

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```
// creating a linked list;
```

```
class Node {
```

```
public:
```

```
    int data;
```

```
    Node* link;
```

```
    // Constructor
```

```
    Node(int n)
```

```
{
```

```
    this->data = n;
```

```
    this->link = NULL;
```

```
}
```

```
};
```

```
class Stack {
```

```
    Node* top;
```

```
public:
```

```
    Stack() { top = NULL; }
```

```
    void push(int data)
```

```
    {
```

```
        // Create new node temp and allocate memory in heap
```

```
        Node* temp = new Node(data);
```

```
        // Check if stack (heap) is full.
```

```
        // Then inserting an element would
```

```
        // lead to stack overflow
```

```
        if (!temp) {
```

```
            cout << "\nStack Overflow";
```

```
            exit(1);
```

```
        }
```

```
        // Initialize data into temp data field
```

```
        temp->data = data;
```

```
        // Put top pointer reference into temp link
```

```
        temp->link = top;
```

```
        // Make temp as top of Stack
```

```

        top = temp;
    }

// Utility function to check if
// the stack is empty or not
bool isEmpty()
{
    // If top is NULL it means that
    // there are no elements are in stack
    return top == NULL;
}

// Utility function to return top element in a stack
int peek()
{
    // If stack is not empty , return the top element
    if (!isEmpty())
        return top->data;
    else
        exit(1);
}

// Function to remove
// a key from given queue q
void pop()
{
    Node* temp;

    // Check for stack underflow

```

```

        if (top == NULL) {
            cout << "\nStack Underflow" << endl;
            exit(1);
        }
        else {
            // Assign top to temp
            temp = top;
            // Assign second node to top
            top = top->link;
            // This will automatically destroy
            // the link between first node and second node
            // Release memory of top node
            // i.e delete the node
            free(temp);
        }
    }

    // Function to print all the
    // elements of the stack
    void display()
    {
        Node* temp;
        // Check for stack underflow
        if (top == NULL) {
            cout << "\nStack Underflow";
            exit(1);
        }
        else {
            temp = top;
            while (temp != NULL) {

```

```

        // Print node data
        cout << temp->data;

        // Assign temp link to temp
        temp = temp->link;

        if (temp != NULL)
            cout << " -> ";

    }

}

};

// Driven Program
int main()
{
    // Creating a stack
    Stack s;

    // Push the elements of stack
    s.push(11);
    s.push(22);
    s.push(33);
    s.push(44);

    // Display stack elements
    s.display();

    // Print top element of stack
    cout << "\nTop element is " << s.peek() << endl;

    // Delete top elements of stack
    s.pop();
    s.pop();

    // Display stack elements
    s.display();

```

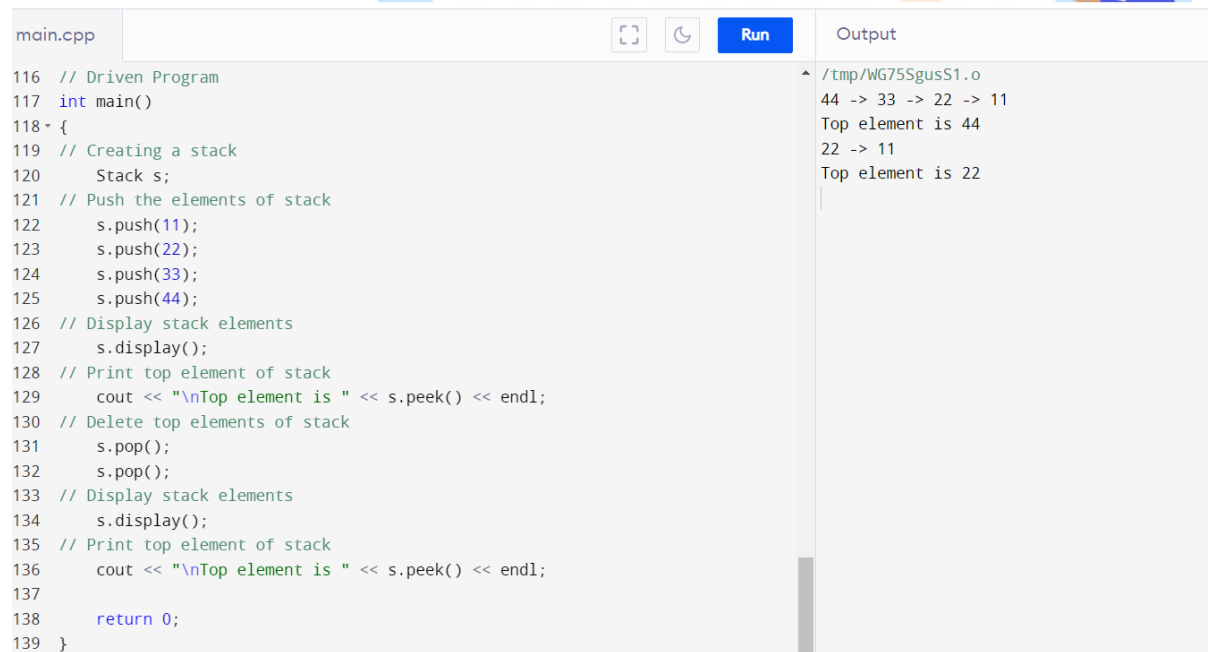
```
// Print top element of stack

    cout << "\nTop element is " << s.peek() << endl;

    return 0;

}
```

Output:



The screenshot shows a C++ IDE with a file named 'main.cpp'. The code implements a stack using a vector. It pushes elements 11, 22, 33, and 44. Then it displays the stack, prints the top element (44), and pops the top two elements (44 and 33). Finally, it displays the stack again and prints the top element (22). The output window on the right shows the execution results: '44 -> 33 -> 22 -> 11', 'Top element is 44', '22 -> 11', and 'Top element is 22'.

```
main.cpp
116 // Driven Program
117 int main()
118 {
119     // Creating a stack
120     Stack s;
121     // Push the elements of stack
122     s.push(11);
123     s.push(22);
124     s.push(33);
125     s.push(44);
126     // Display stack elements
127     s.display();
128     // Print top element of stack
129     cout << "\nTop element is " << s.peek() << endl;
130     // Delete top elements of stack
131     s.pop();
132     s.pop();
133     // Display stack elements
134     s.display();
135     // Print top element of stack
136     cout << "\nTop element is " << s.peek() << endl;
137
138     return 0;
139 }
```

Output

```
/tmp/WG75SgusS1.o
44 -> 33 -> 22 -> 11
Top element is 44
22 -> 11
Top element is 22
```

## 2)Queue using linkedlist:

```
#include<iostream>
```

```
using namespace std;
```

```
struct LinkedListNode //structure of link node
```

```
{
    int data;

    LinkedListNode *next;
} *front = NULL,*rear = NULL,*pointer = NULL,*nextpointer = NULL;
```

```
void enqueue (int item) //push the value in the queue
```

```
{
```

```

nextpointer = new LinkedListNode;
nextpointer->data = item;
nextpointer->next = NULL;
if (front == NULL)
{
    front = rear = nextpointer;
    rear->next = NULL;
}
else
{
    rear->next = nextpointer;
    rear = nextpointer;
    rear->next = NULL;
}
}

int dequeue () //remove the value from the queue
{
    int item;
    if (front == NULL)
    {
        cout << "Queue is empty!!\n";
    }
    else
    {
        pointer = front;
        item = pointer->data;
        front = front->next;
        delete (pointer);
    }
}

```

```

        return (item);
    }
}

int main ()
{

    int n, counter = 0, x1;
    cout << "Enter the number of values to be pushed into queue:-\n";
    cin >> n;
    cout << "Enqueue the value:-\n";
    while (counter < n) { cin >> x1;
        enqueue (x1);
        counter++;
    }
    cout << " After Dequeue :-\n";
    while (true)
    {
        if (front != NULL)
            cout << dequeue () << endl;
        else
            break;
    }
}

```

Output :



```
main.cpp
35     cout << "Queue is empty!!\n";
36 }
37 else
38 {
39     pointer = front;
40     item = pointer->data;
41     front = front->next;
42     delete (pointer);
43     return (item);
44 }
45 }
46
47 int main ()
48 {
49
50     int n, counter = 0, x1;
51     cout << "Enter the number of values to be pushed into queue:-\n";
52     cin >> n;
53     cout << "Enqueue the value:-\n";
54     while (counter < n) { cin >> x1;
55         enqueue (x1);
56         counter++;
57     }
58     cout << " After Dequeue :-\n";
59     while (true)
```

Output

```
/tmp/PDt9V2mZw4.o
Enter the number of values to be pushed into queue:-
3
Enqueue the value:-
25
50
75
After Dequeue :-
25
50
75
```

### 3)Doubly linkedlist:

```
#include <iostream>
```

```
using namespace std;
```

```
struct Node {
```

```
    int data;
```

```
    struct Node *prev;
```

```
    struct Node *next;
```

```
};
```

```
struct Node* head = NULL;
```

```
void insert(int newdata) {
```

```
    struct Node* newnode = (struct Node*) malloc(sizeof(struct Node));
```

```
    newnode->data = newdata;
```

```
    newnode->prev = NULL;
```

```
    newnode->next = head;
```

```
    if(head != NULL)
```

```
        head->prev = newnode ;
```

```
    head = newnode;
```

```
}
```

```
void display() {
```

```

struct Node* ptr;

ptr = head;

while(ptr != NULL) {

    cout<< ptr->data <<" ";

    ptr = ptr->next;

}

}

int main() {

    insert(3);

    insert(1);

    insert(7);

    insert(2);

    insert(9);

    cout<<"The doubly linked list is: ";

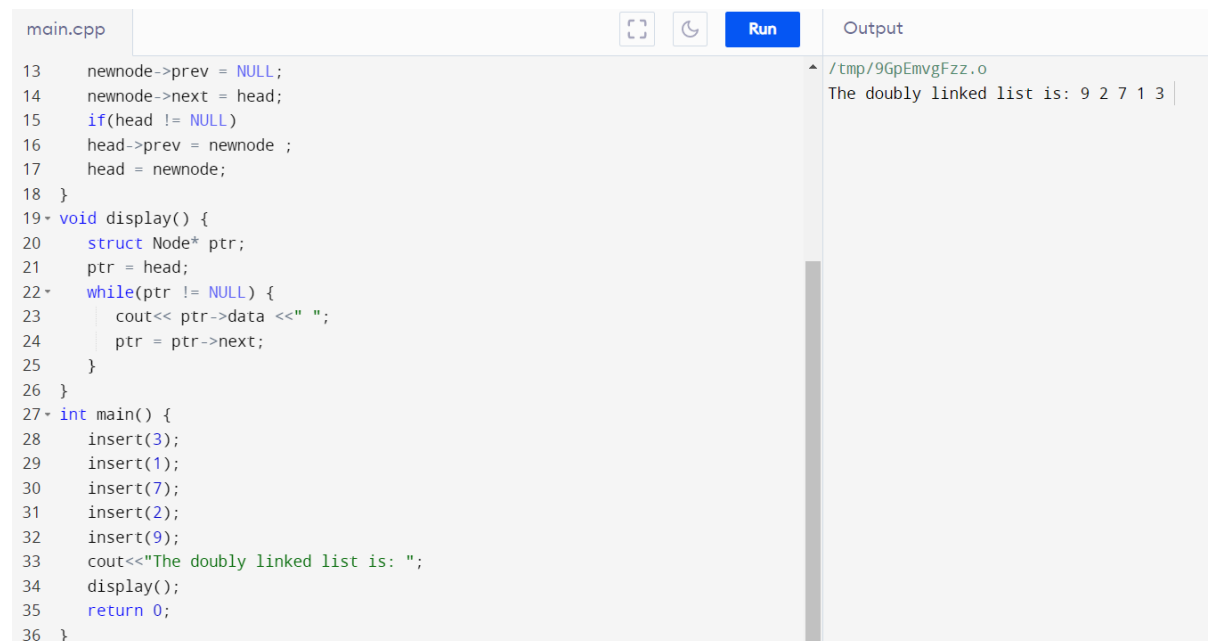
    display();

    return 0;

}

```

## Output:



The screenshot shows a C++ IDE with a file named 'main.cpp'. The code in the editor is as follows:

```

13     newnode->prev = NULL;
14     newnode->next = head;
15     if(head != NULL)
16         head->prev = newnode ;
17     head = newnode;
18 }
19 void display() {
20     struct Node* ptr;
21     ptr = head;
22     while(ptr != NULL) {
23         cout<< ptr->data <<" ";
24         ptr = ptr->next;
25     }
26 }
27 int main() {
28     insert(3);
29     insert(1);
30     insert(7);
31     insert(2);
32     insert(9);
33     cout<<"The doubly linked list is: ";
34     display();
35     return 0;
36 }

```

On the right side, there is an 'Output' panel. It shows the command path '/tmp/9GpEmvgFzz.o' and the output of the program: 'The doubly linked list is: 9 2 7 1 3 |'.

#### 4) Enqueue & Dequeue:

```
#include <iostream>
```

```
using namespace std;
```

```
int queue[50];
```

```
int n = 50;
```

```
int front = - 1;
```

```
int rear = - 1;
```

```
void Queue_insertion() {
```

```
    int val;
```

```
    if (rear == n - 1)
```

```
        cout<<"Queue Overflow"<<endl;
```

```
    else {
```

```
        front = 0;
```

```
        cout<<" insert value in the queue : "<<endl;
```

```
        cin>>val;
```

```
        rear++;
```

```
        queue[rear] = val;
```

```
    }
```

```
}
```

```
void Delete() {
```

```
    if (front == - 1) {
```

```
        cout<<"Queue Underflow ";
```

```
        return ;
```

```
    } else {
```

```
    cout<<"Element deleted from queue is : "<< queue[front] <<endl;
    front++;
}
}
```

```
void Display_Queue () {
    if (front == - 1 )
        cout<<"Queue is empty"<<endl;
    else {
        cout<<"Queue elements are : ";
        for (int i = front; i <= rear; i++)
            cout<<queue[i]<<" ";
        cout<<endl;
    }
}
```

```
int main() {
    int ch;
    cout<<"1) insertion element to the queue"<<endl;
    cout<<"2) Delete element from queue"<<endl;
    cout<<"3) Display all the elements of queue"<<endl;
    cout<<"4) Exit"<<endl;
    do {
        cout<<"Enter your choice : "<<endl;
        cin>>ch;
        switch (ch) {
            case 1: Queue_insertion();
```

```

        break;

case 2: Delete();

        break;

case 3: Display_Queue ();

        break;

case 4: cout<<"Exit"<<endl;

        break;

default: cout<<"Invalid choice"<<endl;

    }

} while(ch!=4);

return 0;

}

```

## Output:

main.cpp	Output
<pre> 38     cout&lt;&lt;queue[1]&lt;&lt; " "; 39     cout&lt;&lt;endl; 40 } 41 } 42 int main() { 43     int ch; 44     cout&lt;&lt;"1) insertion element to the queue"&lt;&lt;endl; 45     cout&lt;&lt;"2) Delete element from queue"&lt;&lt;endl; 46     cout&lt;&lt;"3) Display all the elements of queue"&lt;&lt;endl; 47     cout&lt;&lt;"4) Exit"&lt;&lt;endl; 48     do { 49         cout&lt;&lt;"Enter your choice : "&lt;&lt;endl; 50         cin&gt;&gt;ch; 51         switch (ch) { 52             case 1: Queue_insertion(); 53                 break; 54             case 2: Delete(); 55                 break; 56             case 3: Display_Queue (); 57                 break; 58             case 4: cout&lt;&lt;"Exit"&lt;&lt;endl; 59                 break; 60             default: cout&lt;&lt;"Invalid choice"&lt;&lt;endl; 61         } 62     } while(ch!=4); 63     return 0; 64 } </pre>	<pre> /tmp/NemIST6uZ5.o 1) insertion element to the queue 2) Delete element from queue 3) Display all the elements of queue 4) Exit Enter your choice : 1 insert value in the queue : 10 Enter your choice : 1 insert value in the queue : 24 Enter your choice : 1 insert value in the queue : 39 Enter your choice : 3 Queue elements are : 10 24 39 Enter your choice : 2 Element deleted from queue is : 10 Enter your choice : 4 Exit </pre>