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) Enque
5) Deque

1)Stack using linkedlist:

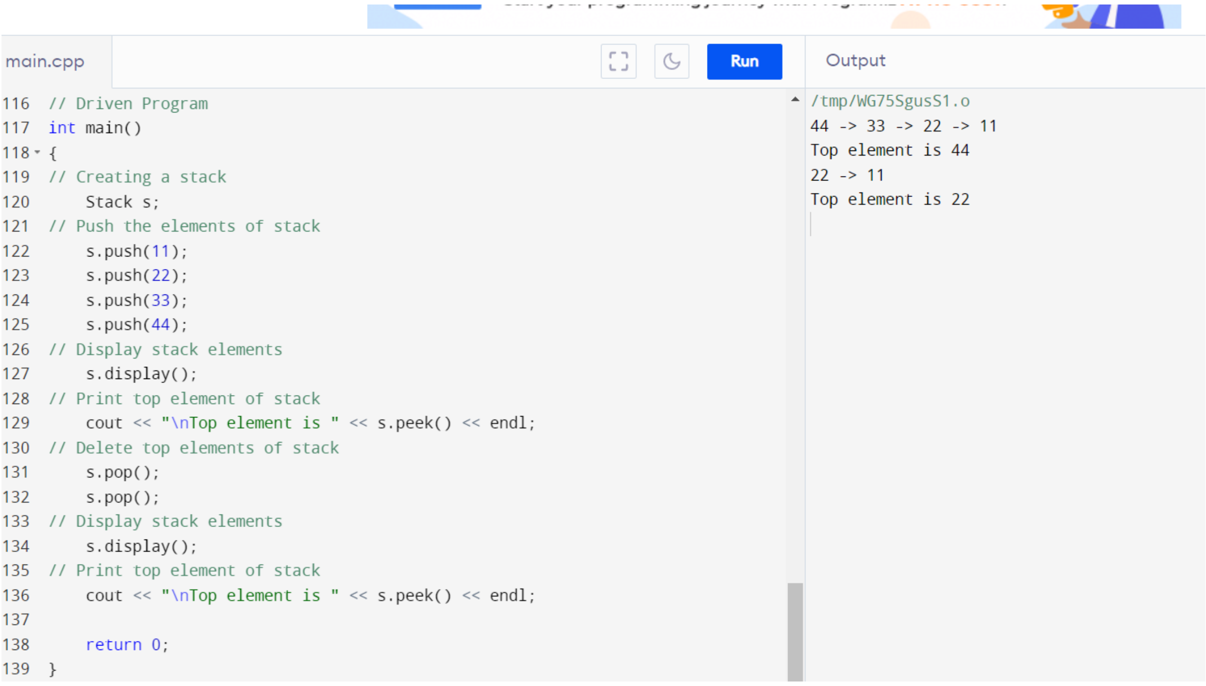
```
// C++ program to Implement a stack

// 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:



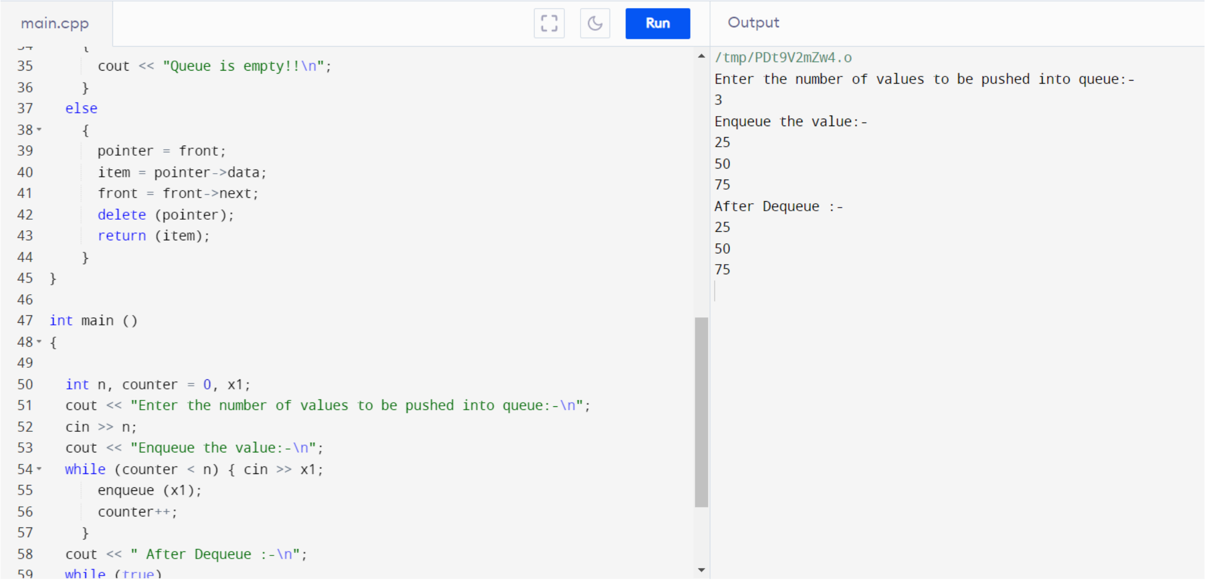
```
// Driven Program
int main()
{
// Creating a stack
    Stack s;
// Push the elements of stack
    s.push(11);
    s.push(22);
    s.push(33);
    s.push(44);
// Display stack elements
    s.display();
// Print top element of stack
    cout << "\nTop element is " << s.peek() << endl;
// Delete top 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;
}
```

```
/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 :



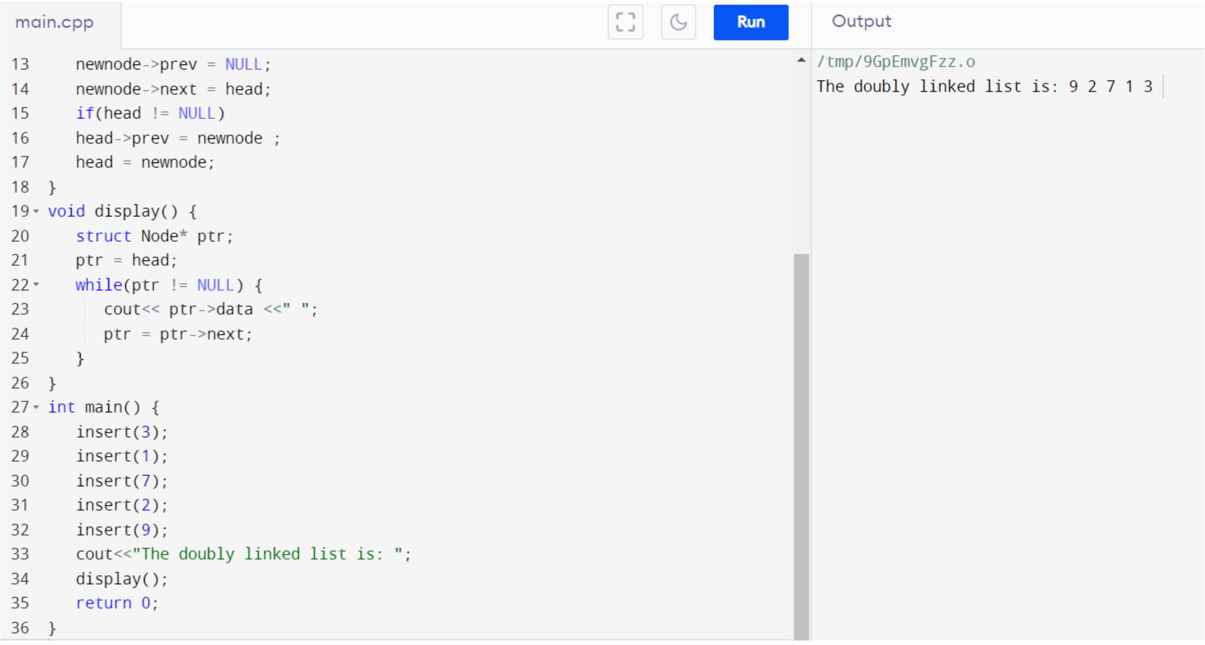
```
    {
      cout << "Queue is empty!!\n";
    }
  else
    {
      pointer = front;
      item = pointer->data;
      front = front->next;
      delete (pointer);
      return (item);
    }
}

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)
```

```
/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:



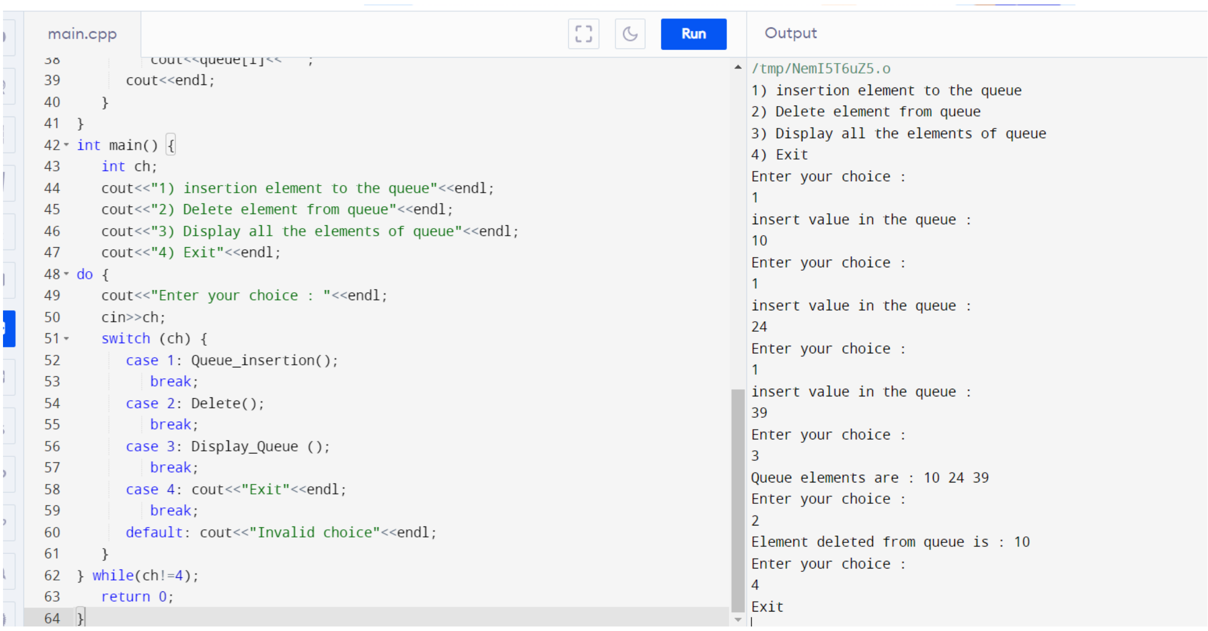
```
   newnode->prev = NULL;
   newnode->next = head;
   if(head != NULL)
   head->prev = newnode ;
   head = newnode;
}
void display() {
   struct Node* ptr;
   ptr = head;
   while(ptr != NULL) {
      cout<< ptr->data <<" ";
      ptr = ptr->next;
   }
}
int main() {
   insert(3);
   insert(1);
   insert(7);
   insert(2);
   insert(9);
   cout<<"The doubly linked list is: ";
   display();
   return 0;
}
```

```
/tmp/9GpEmvgFzz.o
The doubly linked list is: 9 2 7 1 3
```

4)Enque & Deque:

```
#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:



```
            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;
}
```

```
/tmp/NemI5T6uZ5.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
```