WAP (menu based) for **Linked list implementation of STACK**, where each node consist of name & marks of students.

**#include<iostream.h>**

**#include<stdio.h>**

**#include<conio.h>**

**struct node**

**{**

**char name[20];**

**int marks;**

**node \*next;**

**};**

**class stack**

**{**

**node \*top;**

**public:**

**stack(){top=NULL;}**

**void push();**

**void pop();**

**void display();**

**~stack();**

**};**

**void stack::push() //Addition of a node at the top**

**{**

**node \*temp;**

**temp=new node;**

**gets(temp->name);**

**cin>>temp->marks;**

**temp->next=top;**

**top=temp;**

**}**

**void stack::pop() //Deletion of the node at the top**

**{**

**if(top==NULL)**

**cout<<"Underflow!!Stack is empty\n";**

**else**

**{**

**node \*temp;**

**temp=top;**

**top=top->next;**

**delete(temp);**

**}**

**}**

**void stack::display() //Displays the Stack**

**{**

**if(top==NULL)**

**cout<<"Stack is empty\n";**

**else**

**{**

**node \*temp;**

**temp=top;**

**while(temp!=NULL)**

**{**

**cout<<temp->name;**

**cout<<temp->marks;**

**temp=temp->next;**

**}**

**}**

**}**

**stack::~stack() //Destructor**

**{**

**node \*temp;**

**while(top!=NULL)**

**{**

**temp=top;**

**top=top->next;**

**delete(temp);**

**}**

**}**

**void main()**

**{**

**stack s;**

**int ch;**

**do**

**{**

**cout<<"1.PUSH\n2.POP\n3.DISPLAY\n4.QUIT\n";**

**ch=getch();**

**switch(ch)**

**{**

**case 1: s.push();break;**

**case 2: s.pop();break;**

**case 3: s.display();break;**

**case 4: break;**

**default:cout<<"Wrong Choice";**

**}**

**}while(ch!=4);**

**}**

WA menu based program for **Linked list implementation of QUEUE**, where each node consist of name and marks of a student.

**#include<iostream.h>**

**#include<stdio.h>**

**struct node**

**{**

**char name[20];**

**int marks;**

**node \*next;**

**};**

**class queue**

**{**

**node \*front;**

**node \*rear;**

**public:**

**queue(){front=rear=NULL;}**

**void addq();**

**void delq();**

**void display();**

**~queue();**

**};**

**void queue::addq() //Addition of a node at the rear**

**{**

**node \*temp;**

**temp=new node;**

**gets(temp->name);**

**cin>>temp->marks;**

**temp->next=NULL;**

**if(front!=NULL)**

**{ rear->next=temp;**

**rear=temp;**

**}**

**else**

**front=rear=temp;**

**}**

**void queue::delq() //Deletion of the node at the front**

**{**

**if(front==NULL)**

**cout<<"Underflow!!Queue is empty\n";**

**else**

**{ node \*temp;**

**temp=front;**

**front=front->next;**

**delete(temp);**

**if(front==NULL)rear=NULL;**

**}**

**}**

**void queue::display() //Displays the Queue**

**{**

**if(front==NULL)**

**cout<<"Queue is empty\n";**

**else**

**{**

**node \*temp;**

**temp=front;**

**while(temp!=NULL)**

**{**

**cout<<temp->name;**

**cout<<temp->marks;**

**temp=temp->next;**

**}**

**}**

**}**

**queue::~queue() //Destructor**

**{**

**node \*temp;**

**while(front!=NULL)**

**{**

**temp=front;**

**front=front->next;**

**delete(temp);**

**}**

**}**

**void main()**

**{**

**queue q;**

**int ch;**

**do**

**{**

**cout<<"1.ADD\n2.DEL\n3.DISPLAY\n4.QUIT\n";**

**cin>>ch;**

**switch(ch)**

**{**

**case 1: q.addq();break;**

**case 2: q.delq();break;**

**case 3: q.display();break;**

**case 4: break;**

**}**

**}while(ch!=4);**

**}**