NUST COLLEGE OF

ELECTRICAL AND MECHANICAL ENGINEERING

EC 200:DATA STRUCTURE

Lab # 4

Instructor: Anum Abdul Salam

Lab Engineer: Ansa Liaqat

Student's Name: **Furqan Ahmad**

Reg Number: **352076**

Syndicate: **A**

Degree: **42**

Department: **Computer Engineering**

TASk1:

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

template<class T>
class stacks{
T arr[5];
int check;
public:
    stacks();
    void push(T);
    T gettop();
    void pop();
    ~stacks();
};

template<class T>
stacks<T>::stacks(){
    for (int i=0;i<5;i++){
    arr[i]=0;
    }
    check=0;
}

template<class T>
void stacks<T>::push(T x){
    if(check==5){cout<<"Stack overflow\n";
    return;}
    arr[4-check]=x;
    check ++;
}

template<class T>
T stacks<T>::gettop(){
    if(check==0){
        return 0;
    }

    else{
        int x=check-1;
        return(arr[4-x]);}

}
```

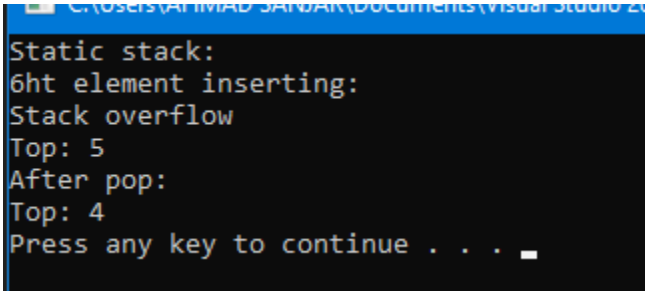
```
template<class T>
void stacks<T>::pop(){
     if(check==0){
         cout<<"Stack empty!\n";
        return;
    }

    else{
        int x=check-1;
        arr[4-x]=0;
    }
    check--;

}

template<class T>
stacks<T>::~stacks(){
   for(int i=0;i<5;i++){
       arr[i]=0;

   }
   check=0;
}
```

MAIN:

```
#include<iostream>
#include"stackdynamic.h"
using namespace std;
int main(){
    stacks <int>a;
    cout<<"Static stack:\n";
    a.push(1);
    a.push(2);
    a.push(3);
    a.push(4);
    a.push(5);
    cout<<"6ht element inserting:\n";
    a.push(6);
    cout<<"Top: "<<a.gettop()<<endl;
    a.pop();
```

```
    cout<<"After pop:\n";
    cout<<"Top: "<<a.gettop()<<endl;

system("pause");
return 0;
}
```

OUTPUT:



TASK 2:

```
template<class T>
struct node{
T value;
node <T>*next;
node(){
next=NULL;}
};

template<class T>
class stackd{
node <T>*head;
public:
    stackd();
    void push(T);
    T gettop();
    void pop();
    stackd(T &a);
    ~stackd();
};

template<class T>
stackd<T>::stackd(){
head=NULL;
```

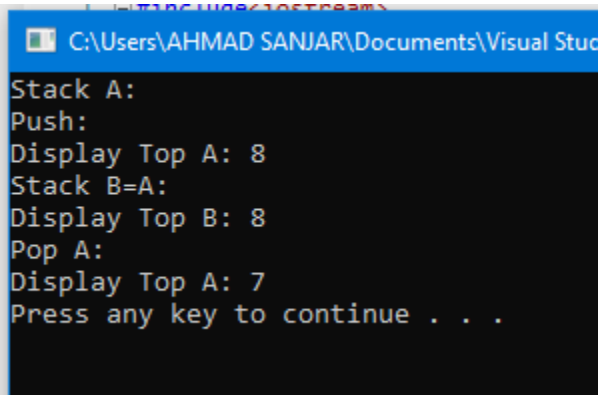
```
}

template<class T>
void stackd<T>::push(T x){
node <T>*a=new node<T>;
a->value=x;
if(head==NULL){
head=a;}
else{
a->next=head;
head=a;
}
}

template<class T>
T stackd<T>::gettop(){
    if(head==NULL){return (T)0;}
    else{return head->value;}
}

template<class T>
void stackd<T>::pop(){
    if(head==NULL){return;}
    else{
    node <T>*temp=head;
    head=head->next;
    delete temp;
    temp=NULL;
    }
}

template<class T>
stackd<T>::stackd(T&a){
head=0;
node <T>*temp2;
node <T>*temp=a.head;
while (temp!=NULL){
    node <T>*x=new node<T>;
    x->value=temp->value;
    if(head==NULL){
        head=x;
        temp2=head;
    }
    else {
        while(temp2->next!=NULL){
```

```
        temp2=temp2->next;
        }
        temp->next=x;
    }
}
}

template<class T>
stackd<T>::~stackd(){
    while(head!=NULL){
    pop();
    }
}
```

MAIN:

```
#include<iostream>
#include"stackdynamic.h"
using namespace std;
int main(){
    stackd <int>a;
    cout<<"Stack A:\n";
    cout<<"Push:\n";
    a.push(5);
    a.push(7);
    a.push(8);
    cout<<"Display Top A: "<<a.gettop()<<endl;
    cout<<"Stack B=A:\n";
    stackd<int>b=a;
    cout<<"Display Top B: "<<b.gettop()<<endl;
    cout<<"Pop A:\n";
    a.pop();
    cout<<"Display Top A: "<<a.gettop()<<endl;




system("pause");
return 0;
}
```

OUTPUT:



TASK3:

```
#include<iostream>
using namespace std;
template<class T>
struct node{
T val;
node<T> *next;
node<T>*previous;
node(){next=NULL;
previous=NULL;}
};

template<class T>
class linkedlist{
node <T>*head;
node <T>*tail;

public:
linkedlist();
linkedlist(linkedlist&);
bool isempty();
void display();
int size();
void IAS(T);
void DAS();
~linkedlist();
```

```
};

template<class T>
linkedlist<T>::linkedlist(){
head=NULL;
tail=NULL;}

template<class T>
bool linkedlist<T>::isempty(){
if(head==NULL){
return true;
}
else{
return false;
}
}

template<class T>
int linkedlist<T>::size(){
int size=0;
node <T>*temp=head;
while(temp!=NULL){
temp=temp->next;
size++;
}
return size;
}

template<class T>
void linkedlist<T>::IAS(T x=NULL){
node <T>*a=new node<T>;
if(x==NULL){
cout<<"Enter Value: ";
cin>>a->val;
}
else{a->val=x;}
if(isempty()){
head=a;
tail=a;
}
else{
a->next=head;
head->previous=a;
head=a;
}
```

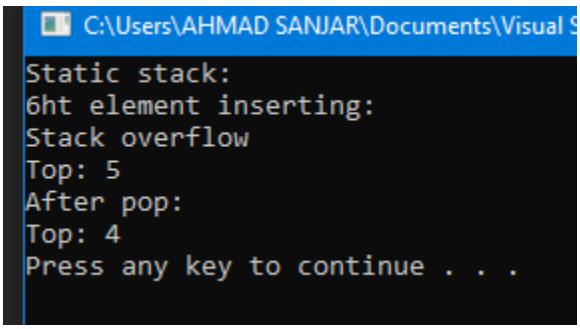
```
}

template<class T>
void linkedlist<T>::display(){

cout<<"Value: "<<head->val<<endl;


}

template<class T>
void linkedlist<T>::DAS(){
if(size()==1){
node <T>*temp=head;
head=NULL;
tail=NULL;
delete temp;
temp=NULL;
return;}
if(isempty()){
cout<<"There is No Node Present!\n";
}
else{
node<T> *temp=head;
head=head->next;
head->previous=NULL;
delete temp;
temp=NULL;
}
}

template<class T>
linkedlist<T>::~linkedlist(){
int x=size();
for(int i=1;i<=x;i++){
DAS();
}
}

template<class T>
linkedlist<T>::linkedlist(linkedlist &a){
head=NULL;
tail=NULL;
if(a.isempty()){return;}
else{
```

```
int size=a.size();
node <T>*tempa=a.head;
for(int i=1;i<=size;i++){
int x=tempa->val;
IAE(x);
tempa=tempa->next;
}
}
}
```

MAIN:

```
#include<iostream>
#include"stackdynamic.h"
using namespace std;
int main(){
    linkedlist <int>a;
    a.IAS(4);
    a.IAS(6);
    cout<<"Top value:\n";
    a.display();
    cout<<"Top value after deleting:\n";
    a.DAS();
    a.display();

system("pause");
return 0;
}
```

OUTPUT:



TASK 1:

A:

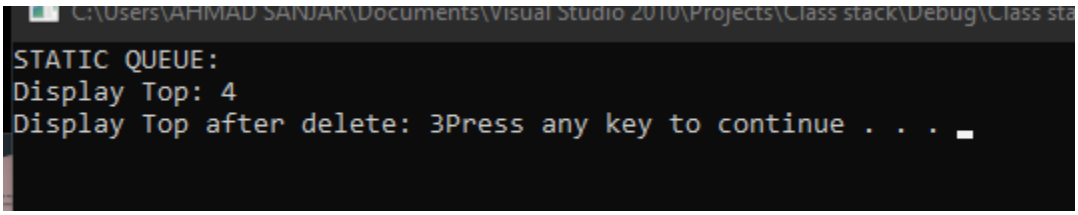
```
#include<iostream>
using namespace std;

template<class T>
class queue{
    int size;
    T   arr[5];
public:
    queue();
    void push(T);
    void gettop();
    void pop();
    ~queue();
};

template<class T>
queue<T>::queue(){
size=0;
for(int i=0;i<5;i++){
arr[i]=0;
}
}

template<class T>
void queue<T>::push(T x){
    if(size<5){
        arr[size]=x;
        size++;
    }
    else{cout<<"Queue if Full!\n";}

}

template<class T>
void queue<T>::gettop(){
    if(size<1){
    cout<<"Queue is empty!\n";
    }
    else{
    cout<< arr[0];
    }
```

```
}

template<class T>
void queue<T>::pop(){
arr[0]=0;
for(int i=0;i<=size;i++){
arr[i]=arr[i+1];
if(i==4){
arr[i]=0;
}
}
size--;

}

template<class T>
queue<T>::~queue(){
for(int i=0;i<5;i++){
arr[i]=0;
}

}
```

MAIN:

```
#include<iostream>
#include"stackdynamic.h"
using namespace std;
int main(){
    queue <int>a;
    cout<<"STATIC QUEUE:\n";
    a.push(4);
    a.push(3);
    a.push(5);
    a.push(7);
    a.push(6);
    cout<<"Display Top: ";
    a.gettop();
    a.pop();
    cout<<"\nDisplay Top after delete: ";
    a.gettop();
system("pause");
return 0;
}
```

OUTPUT:



B:

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

template<class T>
class queue{
    int front,rare,size;
    T   arr[5];
public:
    queue();
    void push(T);
    void gettop();
    void pop();
    ~queue();
};

template<class T>
queue<T>::queue(){
front=rare=size=0;
for(int i=0;i<5;i++){
arr[i]=0;
}
}

template<class T>
void queue<T>::push(T x){
    if(size<5){
        if(rare>4){
            rare=0;
        }
        arr[rare]=x;

        size++;
        rare ++;
```

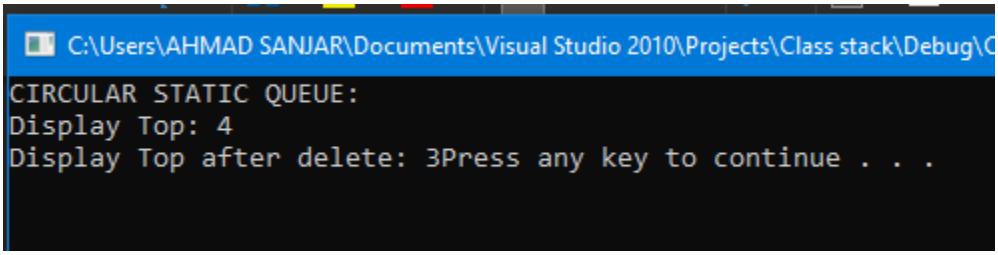
```
    }
    else{cout<<"Queue if Full!\n";}

}

template<class T>
void queue<T>::gettop(){
    if(size<1){
    cout<<"Queue is empty!\n";
    }
    else{
    cout<< arr[front];
    }
}

template<class T>
void queue<T>::pop(){
arr[front]=0;
size--;
if(front<5){
    front++;}
else{
front=0;
}
}

template<class T>
queue<T>::~queue(){
front=rare=size=0;
for(int i=0;i<5;i++){
arr[i]=0;
}
}
```

MAIN:

```
#include<iostream>
#include"stackdynamic.h"
using namespace std;
int main(){
    queue <int>a;
    cout<<"CIRCULAR STATIC QUEUE:\n";
    a.push(4);
    a.push(3);
```

```
    a.push(5);
    a.push(7);
    a.push(6);
    cout<<"Display Top: ";
    a.gettop();
    a.pop();
    cout<<"\nDisplay Top after delete: ";
    a.gettop();
system("pause");
return 0;
}
```

OUTPUT:



TASK 2:

```
#include<iostream>
using namespace std;
template<class T>
struct node{
T val;
node<T> *next;
node<T>*previous;
node(){next=NULL;
previous=NULL;}
};
template<class T>
class linkedlist{
node <T>*head;
node <T>*tail;
public:
linkedlist();
linkedlist(linkedlist&);
bool isempty();
```

```
void display();
int size();
void IAE(T);
void DAS();
~linkedlist();
};

template<class T>
linkedlist<T>::linkedlist(){
head=NULL;
tail=NULL;}

template<class T>
bool linkedlist<T>::isempty(){
if(head==NULL){
return true;
}
else{
return false;
}
}

template<class T>
int linkedlist<T>::size(){
int size=0;
node <T>*temp=head;
while(temp!=NULL){
temp=temp->next;
size++;
}
return size;
}

template<class T>
void linkedlist<T>::IAE(T x=NULL){
node <T>*a=new node<T>;
if(x==NULL){
cout<<"Enter Value: ";
cin>>a->val;
}
else{a->val=x;}
if(this->isempty()){
head=a;
tail=a;
}
```

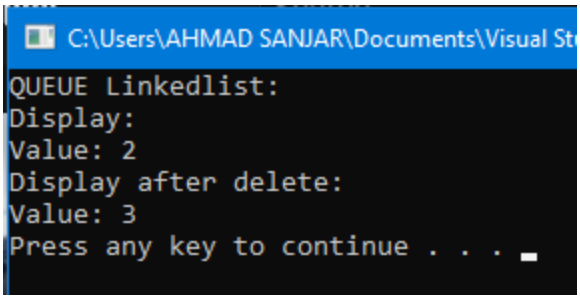
```
else{
a->previous=tail;
tail->next=a;
tail=a;
}
}

template<class T>
void linkedlist<T>::display(){
node <T>*temp=head;
cout<<"Value: "<<temp->val<<endl;

}

template<class T>
void linkedlist<T>::DAS(){
if(size()==1){
node <T>*temp=head;
head=NULL;
tail=NULL;
delete temp;
temp=NULL;
return;}
if(isempty()){
cout<<"There is No Node Present!\n";
}
else{
node<T> *temp=head;
head=head->next;
head->previous=NULL;
delete temp;
temp=NULL;
}
}

template<class T>
linkedlist<T>::~linkedlist(){
int x=size();
for(int i=1;i<=x;i++){
DAS();
}
}

template<class T>
linkedlist<T>::linkedlist(linkedlist &a){
```

```
head=NULL;
tail=NULL;
if(a.isempty()){return;}
else{
int size=a.size();
node <T>*tempa=a.head;
for(int i=1;i<=size;i++){
int x=tempa->val;
IAE(x);
tempa=tempa->next;
}
}
}
```

MAIN:

```
#include<iostream>
#include"stackdynamic.h"
using namespace std;
int main(){
    linkedlist <int> a;
    cout<<"QUEUE Linkedlist:\n";
    a.IAE(2);
    a.IAE(3);
    a.IAE(5);
    a.IAE(6);
    a.IAE(7);
    a.IAE(7);
    cout<<"Display:\n";
    a.display();
    cout<<"Display after delete:\n";
    a.DAS();
    a.display();
system("pause");
return 0;
}
```

OUTPUT:


C:\Users\AHMAD SANJAR\Documents\Visual St
QUEUE Linkedlist:
Display:
Value: 2
Display after delete:
Value: 3
Press any key to continue . . .