NUST COLLEGE OF

ELECTRICAL AND MECHANICAL ENGINEERING

EC 200: DATA STRUCTURE

ASSIGNMENT 1

Instructor: Anum Abdul Salam

Lab Engineer: Ansa Liaquat

Student's Name: **Furqan Ahmad**

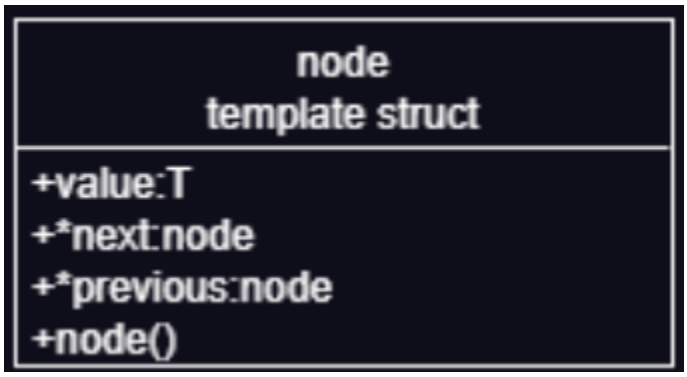
Reg Number: **352076**

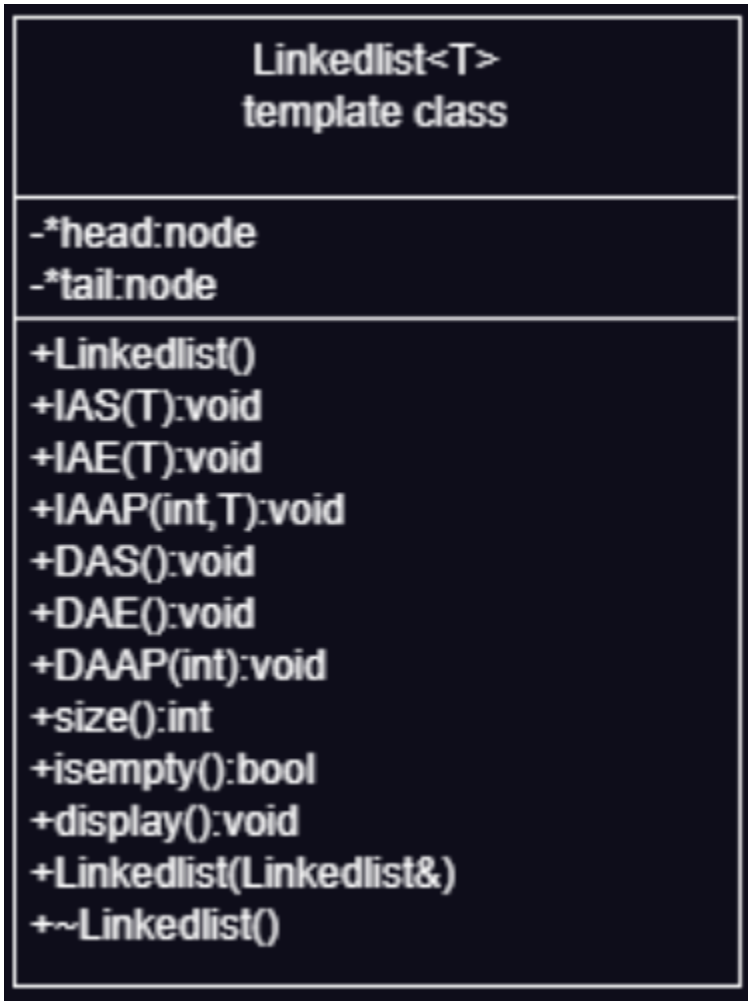
Syndicate: **A**

Degree: **42**

Department: **Computer Engineering**

CLASS UML DAIGRAM:





HEADER.h

```
#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 IAE(T);
    void IAAP(int ,T);
    void DAS();
    void DAE();
    void DAAP(int);
    ~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>::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;
while(temp!=NULL){
cout<<"Value: "<<temp->val<<endl;
temp=temp->next;
}
}

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>
void linkedlist<T>::DAE(){


    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=tail;
    tail=tail->previous;
    tail->next=NULL;
    delete temp;
    temp=NULL;
    }
}

template<class T>
void linkedlist<T>::IAAP(int p=NULL,T x=NULL){
    node <T>*a=new node<T>;
    node <T>*temp=head;
    int check=1;

    if(p==NULL){
    cout<<"Enter Position: ";
    cin>>p;
    }

    if(x==NULL){
    cout<<"Enter Value: ";
    cin>>x;
    }

    if(p<1){
    cout<<"This Position Does not Exist!\n";
    return;
    }
```

```
    if(p==1){
    IAS(x);
    return;
    }

    if(p>size()){
    IAE(x);
    return;
    }
    a->val=x;
    while(temp!=NULL){

        if(p==check){
        temp->previous->next=a;
        a->next=temp;
        a->previous=temp->previous;
        return;

        }

        else{
            check++;
            temp=temp->next;
        }
    }

    cout<<"This Position Does not Exist!\n";
}

template<class T>
void linkedlist<T>::DAAP(int p=NULL){
    node <T>*temp=head;
    int check=1;


    if(p==NULL){
    cout<<"Enter Position: ";
    cin>>p;
    }

    if(p<1){
    cout<<"This Position Does not Exist!\n";
    return;
    }
```

```
    if(p==1){
    DAS();
    return;}

    if(p==size()){
    DAE();
    return;
    }

    while(temp!=NULL){

        if(p==check){
        temp->previous->next=temp->next;
        temp->next->previous=temp->previous;
        //delete temp;
        //temp=NULL;
        return;

        }

        else{
            check++;
            temp=temp->next;
        }
    }
    cout<<"This Position Does not Exist!\n";
}

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.cpp:

```
#include<iostream>

#include"Class.h"
using namespace std;

int main(){

    linkedlist <int>a;
system("pause");
}
```

This is the basic body of main.

Following are the functions that are called in main.

Insert At Start:

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

int main(){

    linkedlist <int>a;
    a.IAS(3);
    a.IAS(2);
    a.IAS(4);
    a.IAS(6);
    a.IAS(9);
    a.display();
system("pause");
}
```

OUTPUT:

```
C:\Users\AHMAD SANJAR\Documents\Visual
Value: 9
Value: 6
Value: 4
Value: 2
Value: 3
Press any key to continue . . .
```

Insert At END:

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

int main(){

    linkedlist <int>a;
    a.IAE(3);
    a.IAE(2);
    a.IAE(4);
    a.IAE(6);
    a.IAE(9);
    a.display();
system("pause");
}
```

OUTPUT:

```
C:\Users\AHMAD SANJAR\Documents\Visual St
Value: 3
Value: 2
Value: 4
Value: 6
Value: 9
Press any key to continue . . .
```

Insert at Any Position:

This function takes 2 arguments

1.Position

2.Value

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

int main(){

    linkedlist <int>a;
    a.IAAP(1,3);
    a.IAAP(-2,7);
    a.IAAP(2,4);
    a.IAAP(3,9);
    a.IAAP(2,3);
    a.IAAP(7,4);

    a.display();
system("pause");
}
```

Output:

```
C:\Users\AHMAD SANJAR\Documents\Visual Studio 2010\Pr
This Position Does not Exist!
Value: 3
Value: 3
Value: 4
Value: 9
Value: 4
Press any key to continue . . .
```

The line “This position does not exit” is when we enter position less the 0;

And if we enter position 0 then it asks the user to enter position.

Delete At Start:

```
#include<iostream>
```

```
#include"Class.h"
using namespace std;

int main(){

    linkedlist <int>a;
    a.IAS(4);
    a.IAS(3);
    a.IAS(8);
    a.IAS(2);
    a.IAS(1);
    a.IAS(7);
    cout<<"BEFORE DELETE: \n";
    a.display();
    cout<<"After DELETE 1: \n";
    a.DAS();
    a.display();
    cout<<"After DELETE 2: \n";
    a.DAS();
    a.display();
    cout<<"After DELETE 3: \n";
    a.DAS();
    a.display();
system("pause");
}
```

OUTPUT:

```
C:\Users\AHMAD SANJAR\Documents\Visual Studio
BEFORE DELETE:
Value: 7
Value: 1
Value: 2
Value: 8
Value: 3
Value: 4
After DELETE 1:
Value: 1
Value: 2
Value: 8
Value: 3
Value: 4
After DELETE 2:
Value: 2
Value: 8
Value: 3
Value: 4
After DELETE 3:
Value: 8
Value: 3
Value: 4
Press any key to continue . . .
```

Delete At End:

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

int main(){

    linkedlist <int>a;
    a.IAS(4);
    a.IAS(3);
    a.IAS(8);
    a.IAS(2);
    a.IAS(1);
    a.IAS(7);
    cout<<"BEFORE DELETE: \n";
    a.display();
    cout<<"After DELETE 1: \n";
    a.DAE();
    a.display();
    cout<<"After DELETE 2: \n";
    a.DAE();
```

```
    a.display();
    cout<<"After DELETE 3: \n";
    a.DAE();
    a.display();
system("pause");
}
```

OUTPUT:

```
C:\Users\AHMAD SANJAR\Documents\Visual Studio 2010\Pr
BEFORE DELETE:
Value: 7
Value: 1
Value: 2
Value: 8
Value: 3
Value: 4
After DELETE 1:
Value: 7
Value: 1
Value: 2
Value: 8
Value: 3
After DELETE 2:
Value: 7
Value: 1
Value: 2
Value: 8
After DELETE 3:
Value: 7
Value: 1
Value: 2
Press any key to continue . . .
```

Delete At Any Position:

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

int main(){

    linkedlist <int>a;
    a.IAS(4);
    a.IAS(3);
    a.IAS(8);
    a.IAS(2);
    a.IAS(1);
    a.IAS(7);
    cout<<"BEFORE DELETE: \n";
    a.display();
    cout<<"After DELETE Position 4: \n";
    a.DAAP(4);
    a.display();
    cout<<"After DELETE Position 2: \n";
    a.DAAP(2);
    a.display();
    cout<<"After DELETE Position 3: \n";
    a.DAAP(3);
    a.display();
    a.DAAP(10);
system("pause");
}
```

OUTPUT:

in last 10 position was not existed.

```
C:\Users\AHMAD SANJAR\Documents\Vis
BEFORE DELETE:
Value: 7
Value: 1
Value: 2
Value: 8
Value: 3
Value: 4
After DELETE Position 4:
Value: 7
Value: 1
Value: 2
Value: 3
Value: 4
After DELETE Position 2:
Value: 7
Value: 2
Value: 3
Value: 4
After DELETE Position 3:
Value: 7
Value: 2
Value: 4
This Position Does not Exist!
Press any key to continue . . .
```

COPY CONSTRUCTORE:

```
#include<iostream>
#include"Class.h"
using namespace std;
template<class T>
linkedlist <T>function(){
linkedlist <int>a;
    a.IAS(4);
    a.IAS(3);
    a.IAS(8);
    a.IAS(2);
    a.IAS(1);
    a.IAS(7);
    cout<<"Display A: \n";
    a.display();
    return a;
}



int main(){

    linkedlist <int>b(function<int>());
    cout<<"\n\nDisplay B(B=A) \n";
    b.display();
system("pause");
}
```

OUTPUT:

```
C:\Users\AHMAD SANJAR\Documents\Visual Stu
Display A:
Value: 7
Value: 1
Value: 2
Value: 8
Value: 3
Value: 4


Display B(B=A)
Value: 7
Value: 1
Value: 2
Value: 8
Value: 3
Value: 4
Press any key to continue . . .
```

Cecking for Float:

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



int main(){

    linkedlist <float>a;
    a.IAS(.4324);
    a.IAS(.543);
    a.IAS(.674);
    a.IAS(.2654);
    a.IAS(.32423);
    a.display();
system("pause");
}
```

OUTPUT:

```
C:\Users\AHMAD SANJAR\Documents\Visual Stu
Value: 0.32423
Value: 0.2654
Value: 0.674
Value: 0.543
Value: 0.4324
Press any key to continue . . .
```