# DS&A LAB TASKS

211042 Noman Masood A

OCTOBER 20, 2022

AIR UNIVERSITY

# LAB TAKS 1

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

class Node{
    private:
        string name;
        Node * next = NULL;
    public:

        Node(string name){
            this->name=name;
        }
        string get_name(){
            return name;
        }
        Node * getnext(){
            return next;
        }
        void setnext(Node * next){
            this->next=next;
        }
        friend linked_list;
};

class linked_list{
    private:
        Node * head;
        Node * currentNode ;
    public:
        linked_list();
    void adding_any_position(string name,int position);
        void removig_any_position(int position);
        void add_node_head(string name);
         void remove_node_head();
        void show_head_node();
        void add_tail(string name);
        void remove_tail();
        void show_current(Node * point);
        void printing_data();
        void traverse();
        Node * next(Node * point);
        int length();
        Node * goto_start(Node * point);
        void main_method();
};
```

```
linked_list::linked_list(){
        head=NULL;
        currentNode=NULL;
}

void linked_list::removig_any_position(int position){
        int len=length() , count=1;

        if (position==1){
            remove_node_head();
        }
        else {
            Node * current=head;
            Node * previous;

        do{
        previous=current;
        current=current->next;
        count++;
        }while(count!=position);

        previous->next=current->next;
        delete current;

            }
        }
 void linked_list::adding_any_position(string name,int position){

        Node * NewPoint= new Node(name);
        // creating new node
        Node * temp = head;
        int count=1;
        int len=length();

        if(position<=len){

            while(count!=position){
                temp = temp->next;
                 // going to next node
                count++;
             }

            NewPoint->next = temp->next;
            temp->next=NewPoint;
            }
    else
    cout<<"\n The Node can not be added at this position";

 }
```

```
}
void linked_list::add_node_head(string name){

        Node * point = new Node(name);
           point->next=head;
           head=point;

    cout<<"\n The Node is succesfly added to the head of the Linked list "<<endl;
}

void linked_list::show_head_node(){
    if(head==NULL){
        cout<<"\n The class head is empty";
    }
    else{
        Node * point = head;
        cout<<"\n Data at the head of the Node is "<<point->get_name()<<endl;
        }
    }
void linked_list::remove_node_head(){

    Node * point = head;
    head=head->next;
    delete point;

    cout<<"\n The node at the head of the node is deleted "<<endl;
}
int linked_list::length(){
    int count=0;

    Node * point = head;
    while(point!=0){
        count++;
        point=point->next;
    }
    return count;
}

void linked_list::remove_tail(){

    Node * current = head;
    Node * previous = head;
    current = current->next;
    while(current->next!=NULL){
        previous = previous->next;
        current  = current->next;
    }
    previous->next=current->next;
    delete current;
}
```

```
void linked_list::add_tail(string name){
    Node * point = head;
    Node * node = new Node(name);

    while(point->next!=NULL){
        point=point->next;
    }
    point->next=node;
    node->next=NULL;

}
void linked_list::printing_data(){
    Node * point = head;
     while( point!=NULL){
        cout<<point->get_name()<<endl;
        point=point->next;
     }
}
void linked_list::show_current(Node * point){
    cout<<"\n THE data of the current Node is "<<point->get_name();
}

Node * linked_list::next(Node * point){
    point=point->next;
    return point;
}
```

```
Node * linked_list::goto_start(Node * point){
    point=head;
    return point;
}

void linked_list::main_method(){

    add_node_head(" Noman masood khan");
    add_node_head(" hassan Raza ");
    add_tail("ali");
    remove_tail();

    adding_any_position("Ali khan",1);
    printing_data();

    removig_any_position(3);
    printing_data();

}
```

```
int main(){

    linked_list * dynamicObj = new linked_list();

    dynamicObj->main_method();

}
```

## Lab Tasks no 2

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

class Node{
    private:
        string name;
        Node * next = NULL;
        Node * previous = NULL;
    public:

        Node(string name){
            this->name=name;
        }
        string get_name(){
            return name;
        }
        Node * getnext(){
            return next;
        }
        void setnext(Node * next){
            this->next=next;
        }
        friend linked_list;
};
```

```
class linked_list{
    private:
        Node * head = NULL;
        Node * tail = NULL;
    public:
        void show_head_node(){
        cout<<endl<<head->get_name()<<endl;
        }

        void add_node_head(string data){
            // creating new node
            Node * NewNode = new Node(data);
            if(head==NULL){

                head=NewNode;
                tail=NewNode;
            }
            else{
            head->previous=NewNode;
            NewNode->next=head;
            head=NewNode;
        }
    }
        void remove_node_head(){
            int len = length();
            if(head==NULL){
                cout<<"\n The Head Node cannot be deleted ";
            }
            else if(len==1){
                head=NULL;
                tail=NULL;
            }
            else{
                Node * temp=head;
                head=head->next;
                head->previous=temp->previous;

                delete temp;
            }
        }

        void add_node_tail(string data){
            Node * NewNode = new Node(data);
            if(tail==NULL){
                tail=NewNode;
                head=NewNode;
        }
            tail->next=NewNode;
            NewNode->previous=tail;
            tail=NewNode;
        }
```

```
        void removing_node_position(int position){
        int len=length();

        if(position>len)
        cout<<"\n Node cannot be removed from this position ";
        else{

            if(position==1)
                remove_node_head();
            else{
                    Node * temp = head;
                     int count=1;

                    while(temp!=NULL&&count!=position){
                        temp=temp->next;
                        count++;
                     }

                     temp->previous->next=temp->next;
                     temp->next->previous=temp->previous;
                     delete temp;

            }

        }
    }
```

```
    void add_node_position(int position,string data){
        int len=length();

        if(len<position)
            cout<<"\n The Data cannot be added at this position "<<endl;
        else{

            if(position==1)
                add_node_head(data);

            else{

                int count = 1;
                Node * point = head ;

                while(point!=NULL&&count!=position){
                    point=point->next;
                    count++;
        }
                Node * newNode = new Node(data);

                newNode->next=point->next;
                point->next->previous=newNode;
                point->next=newNode;
                newNode->previous=point;
    }
        }}
```

```
int length(){
    Node * pointer = head;
    int count=0;

    while ( pointer != NULL){
            count++;
            pointer=pointer->next;
}
    return count;
}

void printing_all_nodes(){

    if(head==NULL)
        cout<<" The Data cannot be printed "<<endl;

    else{
    Node * temp = head;

    while(temp!=NULL){
        cout<<temp->get_name()<<endl;
        temp=temp->next;

}}}

void current_node_element(Node * point){
    cout<<point->get_name();
}
```

```
    void next_node(Node *  hnext){
        hnext=hnext->next;
        return hnext;
    }

    void main_method(){
        add_node_tail("Noman");
        add_node_tail("Masood");
        add_node_head("MR.");
        remove_node_head();
        remove_node_tail();
        current_node_element(Node * point);
        add_node_position(2,"Cyber Student");
        removing_node_position(1);
        show_head_node();
        printing_all_nodes();
}
int main(){
    linked_list * obj = new linked_list();
    obj->main_method();
}
```