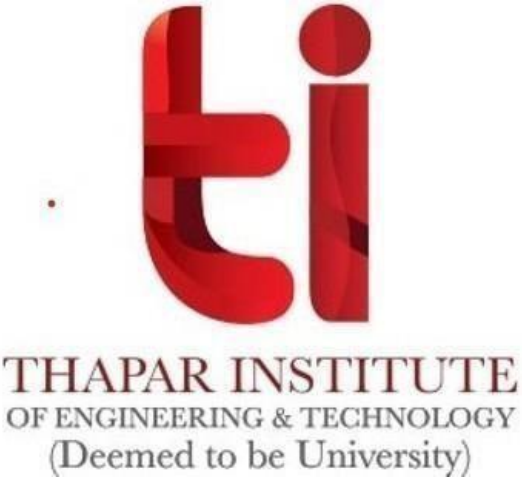
A Practical Activity Report For
Data Structures and Algorithms (UCS406)

Submitted By: **Vivek Arora**

101715178
(ENC 8)

Submitted To:

Dr. Sanjay Sharma



ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR INSTITUTE OF ENGINEERING &TECHNOLOGY, (DEEMED TO BEUNIVERSITY), PATIALA, PUNJAB

ASSIGNMENT 5

QUESTION 1(Various Functions of Array)

i. Display the elements of a Linked list
ii. Count and sum the struct Nodes of a LL
iii. Search for a key element in a LL
iv. Delete an element from a LL
v. Check if a LL is sorted
vi. Merge 2 LLs
vii. Concatenate 2 LLs
viii. Reverse the elements of a LL
ix. Create and Display a circular LL
x. Create a Doubly LL, insert a doubly LL and reverse a doubly LL

i-viii

```
#include<iostream>
using namespace std;
class node
{
   public:
   int data;
   node* next;
   node(int d)
   {
      data=d;
      next=NULL;
   }
};
class linklist
{
   public:
   node* head ;
   node* tail;
    linklist()
   {
    head=tail=NULL;
 }

void create(int data)
```

```
{

    if(head==NULL)
   {
      node* n=new node(data);
      head=tail=n;
   }
   else{
      node* n=new node(data);
      tail->next=n;
      tail=n;
   }
}

void displaylinkedlist()
   { node* temp=head;
      while(temp!=NULL)
      {cout<<temp->data<<" ==> ";
      temp=temp->next;}
}

void countandsumofLL()
{
     int c=0,sum=0;
    node* temp=head;
    while(temp!=NULL)
      {
         c++;
        sum=sum+temp->data;
        temp=temp->next;
       }
cout<<c<<" & "<<sum<<endl;
}

void searchkeyelementinaLL(int n)
{       int c=0;
       node* temp=head;
       while(temp!=NULL)
       {c++;
       if (temp->data==n)
          cout<<"element is present at node"<<c<<endl;
       temp=temp->next;}
}
```

```
void deleteathead()
{
     node* temp=head;
    head=head->next;
    delete temp;
    cout<<"deletion of node at head give us the link list as"<<endl;
    displaylinkedlist();
}

void deleteattail()
{
     node* prev=NULL;
     node* temp=head;
     while(temp->next!=NULL)
     {
         prev=temp;
        temp=temp->next;
     }
     delete temp;
     prev->next=NULL;
     temp=prev;
    cout<<"deletion of node  at tail give us the link list as"<<endl;
    displaylinkedlist();
}

void deleteatmid(int n)
{
    node* temp=head;
    int c=1;
    while(c<=n-2)
    {
        temp=temp->next;
        c++;
     }
    node* present=temp->next;
   temp->next=present->next;
    delete present;
   cout<<"deletion of node give us the link list as"<<endl;
   displaylinkedlist();
}
```

```
void sortedornot()
{
       node* temp=head;
    while(temp!=NULL)
    {
        if (temp->data > temp->next->data)
       {
       cout<<endl<<"the linklist is not sorted"<<endl;
       break;
        }
       temp=temp->next;
     }
}

void reverselinkedlist()
{
       node* temp=head;
       node* prev=NULL,*next=NULL;
      while(temp!=NULL)
      {
          next=temp->next;
          temp->next=prev;
          prev=temp;
          temp=next;}
          head=prev;
          displaylinkedlist();
      }

void concatenatelinkedlist(linklist l2)
   {
     node* temp=head;
     while(temp->next!=NULL)
          temp=temp->next;
          temp->next=l2.head;
           displaylinkedlist();
        }
   };

int main()
{
   int n,a;
```

```
    cin>>n;
    linklist l;
    linklist l2;
    cout<<"input data n times "<<endl;
    for(int i=0;i<n;i++)
       {
        cin>>a;
        l.create(a);
       }
    cout<<"the elements of link list are"<<endl;
    l.displaylinkedlist();
    cout<<endl<<"no of elements in linklist and sum are:"<<endl;
    l.countandsum();
    cout<<"enter the element u want to search"<<endl;
    cin>>n;
    l.searchkey(n);
    cout<<endl;
    l.deleteathead();
    cout<<endl;
    l.deleteattail();
    cout<<endl;
    cout<<endl<<"enter the node u want to delete"<<endl;
    cin>>n;
    l.deleteatmid(n);
    l.sortedornot();
    cout<<endl<<"the reversed link list is"<<endl;
    l.reverselinkedlist();
    cout<<endl<<"enter n for another l2"<<endl;
    cin>>n;
    cout<<"input data n times "<<endl;
    for(int i=0;i<n;i++)
    {
       cin>>a;
       l2.create(a);
    }
    cout<<"the elements of 2nd link list are"<<endl;
    l2.displaylinkedlist();
    cout<<endl<<"the concatenated link list is"<<endl;
    l.concatenatelinkedlist(l2);
}
```

ix(Circular Linked List)

```
#include<iostream>
using namespace std;
class node
{
   public:
   int data;
   node* next;
   node(int d)
   {
     data=d;
     next=NULL;
   }
};
class linklist
{
   public:
   node* head ;
   node* tail;
   linklist()
    {
     head=tail=NULL;
    }

void create(int data)
{
   if(head==NULL)
   {
      node* n=new node(data);
      head=tail=n;
   }
   else {
      node* n=new node(data);
      tail->next=n;
      tail=n;
      if (tail->next==NULL)
      {
       tail->next=head;
      }
   }
}
```

```
void displaylinkedlist()
    {
       node* temp=head->next;
       cout<<head->data<<" ==> ";
       while(temp!=head)
       {
       cout<<temp->data<<" ==> ";
       temp=temp->next;}
       }
     };

int main()
{
    int n,a;
    cin>>n;
    linklist l;
    linklist l2;
    cout<<"input data n times "<<endl;
    for(int i=0;i<n;i++)
    {
       cin>>a;
       l.create(a);
    }
    cout<<"the elements of link list are"<<endl;
    l.displaylinkedlist();
}
```

X(Doubly Linked List)

```
#include<iostream>
using namespace std;
class node
{
    public:
    int data;
    node* next;
    node* prev;
    node(int d)
    {
     data=d;
       next=NULL;}
};
```

```
class linklist
{
   public:
   node* head ;
   node* tail;
   linklist()
   {
    head=tail=NULL;
   }
void create(int data)
{
   if(head==NULL)
   {
      node* n=new node(data);
      n->prev=NULL;
      head=tail=n;
   }
   else{
      node* n=new node(data);
      n->prev=tail;
      tail->next=n;
      tail=n;
   }
}
void displaylinkedlist()
   {
      node* temp=head;
      while(temp!=NULL)
      {cout<<temp->data<<" <==> ";
      temp=temp->next;}
}

void reversell()
{
      node* temp=head;
       node* prev=NULL,*next=NULL;
      while(temp!=NULL)
      {
       next=temp->next;
      temp->next=prev;
      prev=temp;
      temp=next;}
      head=prev;
```

```
      displaylinkedlist();
}

void insertathead(int a)
{
   node* temp=head;
  node* new_node=new node(a);
  new_node->data=a;
   new_node->next=head;
    new_node->prev = NULL;
   head=new_node;
  displaylinkedlist();
}

void insertattail(int a)
{
   node* tail=head;
  node* new_node=new node(a);
  new_node->data=a;
  new_node->next=NULL;
  while(tail->next!=NULL)
    {
     tail=tail->next;}
    tail->next = new_node;
    new_node->prev = tail;
    displaylinkedlist();
}

void insertatmid(int a,int n)
{
    node* temp=head;
    int c=1;
    while(c<=n-2)
    {
        temp=temp->next;
        c++;
     }
    node* new_node=new node(a);
    new_node->next=temp->next;
    temp->next=new_node;
    cout<<"deletion of node give us the link list as"<<endl;
displaylinkedlist();
}
```

```
};
int main()
{
   int n,a;
   cin>>n;
   linklist l;
   linklist l2;
   cout<<"input data n times "<<endl;
   for(int i=0;i<n;i++)
   {cin>>a;
      l.create(a);
   }
   cout<<"the elements of doubly link list are"<<endl;
   l.displaylinkedlist();
    cout<<endl;
    cout<<endl<<"the reversed link list is"<<endl;
   l.reversell();
   cout<<endl;
   cout<<endl<<"enter the element u want to insert "<<endl;
   cin>>a;
   cout<<endl<<"the new linklist is"<<endl;
   l.insertathead(a);
   cout<<endl<<"enter the element u want to insert "<<endl;
   cin>>a;
   cout<<endl<<"the new linklist is"<<endl;
   l.insertattail(a);
   cout<<endl<<"enter the element n prev node u want to insert "<<endl;
   cin>>a>>n;
   cout<<endl<<"the new linklist is"<<endl;
   l.insertatmid(a,n);

}
```