ASSIGNMENT 6

NAME: Ujjwal Pant
ROLL NUMBER: 1024030370

Q1

```
#include<iostream>
using namespace std;
struct Node{int data;Node*next;Node*prev;};
class DoublyList{
Node*head;
public:
DoublyList(){head=NULL;}
void insertFirst(int val){
Node*t=new Node{val,NULL,NULL};
if(!head)head=t;
else{t->next=head;head->prev=t;head=t;}
}
void insertLast(int val){
Node*t=new Node{val,NULL,NULL};
if(!head){head=t;return;}
Node*p=head;
while(p->next)p=p->next;
p->next=t;t->prev=p;
}
void insertAfter(int key,int val){
Node*p=head;
while(p&&p->data!=key)p=p->next;
if(!p){cout<<"Node not found!\n";return;}
```

```
Node*t=new Node{val,p->next,p};
if(p->next)p->next->prev=t;
p->next=t;
}
void insertBefore(int key,int val){
if(!head)return;
if(head->data==key){insertFirst(val);return;}
Node*p=head;
while(p&&p->data!=key)p=p->next;
if(!p){cout<<"Node not found!\n";return;}
Node*t=new Node{val,p,p->prev};
p->prev->next=t;p->prev=t;
}
void deleteNode(int key){
if(!head)return;
Node*p=head;
while(p&&p->data!=key)p=p->next;
if(!p){cout<<"Node not found!\n";return;}
if(p==head)head=head->next;
if(p->prev)p->prev->next=p->next;
if(p->next)p->next->prev=p->prev;
delete p;cout<<"Node deleted.\n";
}
bool search(int key){
Node*p=head;
while(p){if(p->data==key)return true;p=p->next;}
return false;
}
void display(){
Node*p=head;
if(!p){cout<<"List empty\n";return;}
cout<<"Doubly List: ";
```

```
while(p){cout<<p->data<<" ";p=p->next;}
cout<<endl;
}
};
class CircularList{
Node*head;
public:
CircularList(){head=NULL;}
void insertFirst(int val){
Node*t=new Node{val,NULL,NULL};
if(!head){head=t;t->next=head;return;}
Node*p=head;
while(p->next!=head)p=p->next;
t->next=head;p->next=t;head=t;
}
void insertLast(int val){
Node*t=new Node{val,NULL,NULL};
if(!head){head=t;t->next=head;return;}
Node*p=head;
while(p->next!=head)p=p->next;
p->next=t;t->next=head;
}
void insertAfter(int key,int val){
if(!head)return;
Node*p=head;
do{
if(p->data==key){
Node*t=new Node{val,p->next,NULL};
p->next=t;return;
}
p=p->next;
}while(p!=head);
```

```
cout<<"Node not found!\n";
}
void insertBefore(int key,int val){
if(!head)return;
if(head->data==key){insertFirst(val);return;}
Node*p=head;
while(p->next!=head&&p->next->data!=key)p=p->next;
if(p->next==head){cout<<"Node not found!\n";return;}
Node*t=new Node{val,p->next,NULL};
p->next=t;
}
void deleteNode(int key){
if(!head)return;
Node*p=head;Node*pr=NULL;
if(head->data==key){
while(p->next!=head)p=p->next;
Node*d=head;
if(head==p)head=NULL;
else{p->next=head->next;head=head->next;}
delete d;cout<<"Node deleted.\n";return;
}
p=head;
do{
if(p->next->data==key){
Node*d=p->next;p->next=d->next;delete d;cout<<"Node
deleted.\n";return;
}
p=p->next;
}while(p!=head);
cout<<"Node not found!\n";
}
bool search(int key){
```

```
if(!head)return false;
Node*p=head;
do{if(p->data==key)return true;p=p->next;}while(p!=head);
return false;
}
void display(){
if(!head){cout<<"List empty\n";return;}
Node*p=head;
cout<<"Circular List: ";
do{cout<<p->data<<" ";p=p->next;}while(p!=head);
cout<<endl;
}
};
int main(){
DoublyList d;CircularList c;
int ch,t,val,key;
while(true){
cout<<"\n1.Doubly\n2.Circular\n3.Exit\n";cin>>t;
if(t==3)break;
cout<<"1.IF\n2.IL\n3.IA\n4.IB\n5.D\n6.S\n7.Disp\n8.Back\n";cin>>ch;
switch(ch){
case 1:cin>>val;if(t==1)d.insertFirst(val);else c.insertFirst(val);break;
case 2:cin>>val;if(t==1)d.insertLast(val);else c.insertLast(val);break;
case 3:cin>>key>>val;if(t==1)d.insertAfter(key,val);else
c.insertAfter(key,val);break;
case 4:cin>>key>>val;if(t==1)d.insertBefore(key,val);else
c.insertBefore(key,val);break;
case 5:cin>>key;if(t==1)d.deleteNode(key);else
c.deleteNode(key);break;
case
6:cin>>key;cout<<((t==1)?d.search(key):c.search(key)?"Found\n":"Not
Found\n");break;
```

```
case 7:if(t==1)d.display();else c.display();break;
case 8:continue;
default:cout<<"Invalid\n";
}
}
return 0;
}
```

Q2

```
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;
class ListNode{
    public:
    int val;
    ListNode* next;
    ListNode(int val1){
        val=val1;
        next=nullptr;
    }

    void printallelemnt(ListNode* head){
        if(head==NULL) return;
        if(head->next==head) {cout<<head->val<<"\t"<<head->val;
return; }
```

```
        ListNode* temp=head->next;
        cout<<head->val<<"\t";
        while(temp!=head){
            cout<<temp->val<<"\t";
            temp=temp->next;
        }
        cout<<head->val;
        return;

    }

};
int main()
{

    return 0;
}
```

Q3

i)

```
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;
class ListNode{
    public:
    int val;
    ListNode* next;
```

```
    ListNode* prev;
    ListNode(int val1){
        next=nullptr;
        prev=nullptr;
        val=val1;
    }
    int size(ListNode* head){
        int cnt=0;
        if(head==NULL) return 0;
        if(head->next==NULL) return 1;
        ListNode* temp=head;
        while(temp){
            cnt++;
            temp=temp->next;
        }
        return cnt;
    }


};
int main()
{

    return 0;
}
```

ii)

```
#include <iostream>
```

```
#include <iomanip>
#include <string>
using namespace std;
class ListNode{
    public:
    int val;
    ListNode* next;
    ListNode* prev;
    ListNode(int val1){
        val=val1;
        next=nullptr;
        prev=nullptr;
    }
    int size(ListNode* head){
        if(head==NULL) return 0;
        if(head->next==head) return 1;
        int cnt=1;
        ListNode* temp=head->next;
        while(temp!=head){
            cnt++;
            temp=temp->next;
        }
        return cnt;
    }

};
int main()
{

    return 0;
}
```

Q4

```
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;
class ListNode{
    public:
    int val;
    ListNode *next;
    ListNode *prev;
    ListNode(int val1){
        val=val1;
        next=nullptr;
        prev=nullptr;
    }
    ListNode(int val1, ListNode* next1,ListNode* prev1){
        val=val1;
        next=next1;
        prev=prev1;
    }
    bool ispalindrome(ListNode* head){
        if(head==NULL or head->next==NULL) return true;
        ListNode* temp=head;
        while(temp->next!=nullptr){
            temp=temp->next;
        }
```

```
        while(head!=temp && temp->next!=head){
            if(temp->val!=head->val) return false;
            temp=temp->prev;
            head=head->next;
        }
        return true;
    }
};
int main()
{

    return 0;
}
```

Q5

```
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;
class ListNode{
    public:
    int val;
    ListNode *next;
    ListNode(int val1){
        val=val1;
        next=nullptr;
    }
```

```
    ListNode(int val1, ListNode* next1){
        val=val1;
        next=next1;
    }
    bool iscircular(ListNode* head){
        if(head==NULL || head->next==NULL) return false;
        if(head->next==head) return true;
        ListNode* slow=head;
        ListNode* fast=head;
        while(fast){
            slow=slow->next;
            fast=fast->next->next;
            if(fast==slow) return true;
        }
        return false;
    }

};
int main()
{

    return 0;
}
```