“Final PROJECT”

COURSE :

DATA STURCTURES

SUBMITTED TO :

SIR RSHAN

SUBMITTED BY :

Rabia Batool (2021-BSE-064)

SECTION :

B

Topic :

Doubly circular linklist

```
// Online C++ compiler to run C++ program online
#include<iostream>
using namespace std;
class CList
{
private:
struct node
{int data;
node *next;
node *pre;
} *head;
public:
CList(){
head=NULL;
}
bool emptyList(){
if(head==NULL){
return true;
}
else
return false;
}
void insert (int pos, int value){
node *temp=head;
```

```
node *t=new node;
t->data=value;
for (int i = 1;i < pos;i++)
{
temp = temp->next;
if (temp == NULL)
{
cout<<"There are less than ";
cout<<pos<<" elements."<<endl;
return;
}
}
t->next=temp->next;
temp->next->pre=t;
temp->next=t;
t->pre=temp;
}
void insert_begin(int value) {
 node *temp = new node;
 temp->data = value;
 if (head == NULL) {
 head = temp;
 head->next = temp;
 head->pre = temp;
 }
```

```
else {
 temp->next = head;
 temp->pre = head->pre;
 head->pre->next = temp;
 head->pre = temp;
 head = temp;
 }
}
void insert_end(int value){
node *temp = new node;
 temp->data = value;
if (head == NULL) {
 head = temp;
 head->next = temp;
 head->pre = temp;
 }
else{
temp->next = head;
 temp->pre = head->pre;
 head->pre->next = temp;
 head->pre = temp;
}
}
void delete_begin(){
node *temp,*temp1;
```

```
temp=head;
if(head==NULL){
cout<<"\nList is empty!\n";
return;
}
if(head->next==head){
head=NULL;
delete temp;
return;
}
temp1=head;
while(temp1->next!=head)
temp1=temp1->next;
head=temp->next;
head->pre=temp1;
temp1->next=head;
delete temp;
}
void delete_end(){
node *temp=head;
if (temp == NULL) {
cout << "\n List does not exist";
return;
}
if (head->next == head) {
```

```
delete temp;
head = NULL;
return;
}
else{
node*s=head->pre;
head->pre=s->pre;
s->pre->next=head;
delete s;
}
}
void deletex(int value){
if(head->data==value){
delete_begin();
}
else if(head->pre->data==value){
delete_end();
}
else{
node * temp=head->next;
while(temp->next!=head){
if(temp->data==value){
temp->pre->next=temp->next;
temp->next->pre=temp->pre;
return;
```

```
}
else
temp=temp->next;
}
}
}
void traverse() {
 if (head == NULL) {
 cout << "List is empty." << endl;
 return;
 }
 node *s1 = head;
do{
cout<<s1->data<<" ";
s1=s1->next;
}while(s1!=head);
cout<<endl;
}
int max(){
node *max=head;
node *temp=head->next;
do{
if(temp->data>max->data){
max=temp;
}
```

```
temp=temp->next;
}while(temp!=head);
return max->data;
}
void two_node_list(){
node *temp2=head->next;
head->next=head->pre;
head->pre->pre=head;
while(temp2!=head->pre){
node *temp=temp2;
temp2=temp2->next;
delete temp;
}
}
void traverse2(){
if (head == NULL) {
 cout << "List is empty." << endl;
 return;
 }
node *s1 = head->pre;
do{
cout<<s1->data<<" ";
s1=s1->pre;
}while(s1!=head->pre);
cout<<endl;
```

```
}
};
int main()
{
CList c1;
cout<<"Insertion at begin:\n";
c1.insert_begin(1);
 cout<<"Insertion at begin:\n";
c1.insert_begin(2);
c1.traverse();
cout<<"Insert at end:\n";
c1.insert_end(3);
c1.traverse();
cout<<"Insertion after position 2:\n";
c1.insert(2,6);
c1.traverse();
cout<<"Insert after position 3:\n";
c1.insert(3,196);
c1.traverse();
cout<<"Maximum number: \n";
cout<<c1.max()<<endl;
cout<<"Backward traversal: \n";
c1.traverse2();
cout<<"Delete at begin: \n";
c1.delete_begin();
```

```
c1.traverse();

cout<<"Delete at end: \n";

c1.delete_end();

c1.traverse();

cout<<"Delete 6: \n";

c1.deletex(6);

c1.traverse();

cout<<"Delete all nodes except first and last: \n";

c1.two_node_list();

c1.traverse();

return 0;

}
```

Output:

Output

```
/tmp/m47xDX1a7I.o
Insertion at begin:
Insertion at begin:
2 1
Insert at end:
2 1 3
Insertion after position 2:
2 1 6 3
Insert after position 3:
2 1 6 196 3
Maximum number:
196
Backward traversal:
3 196 6 1 2
Delete at begin:
1 6 196 3
Delete at end:
1 6 196
Delete 6:
1 196
Delete all nodes except first and last:
1 196
```