Assignment No. 02

Submitted By	Tayba Asghar
Registration No.	SP22-BCS-077
Section	Section-B
Subject	DSA
Tutor	Mam Yasmeen Jana

COMSATS UNIVERSITY ISLAMABAD, VEHARI CAMPUS

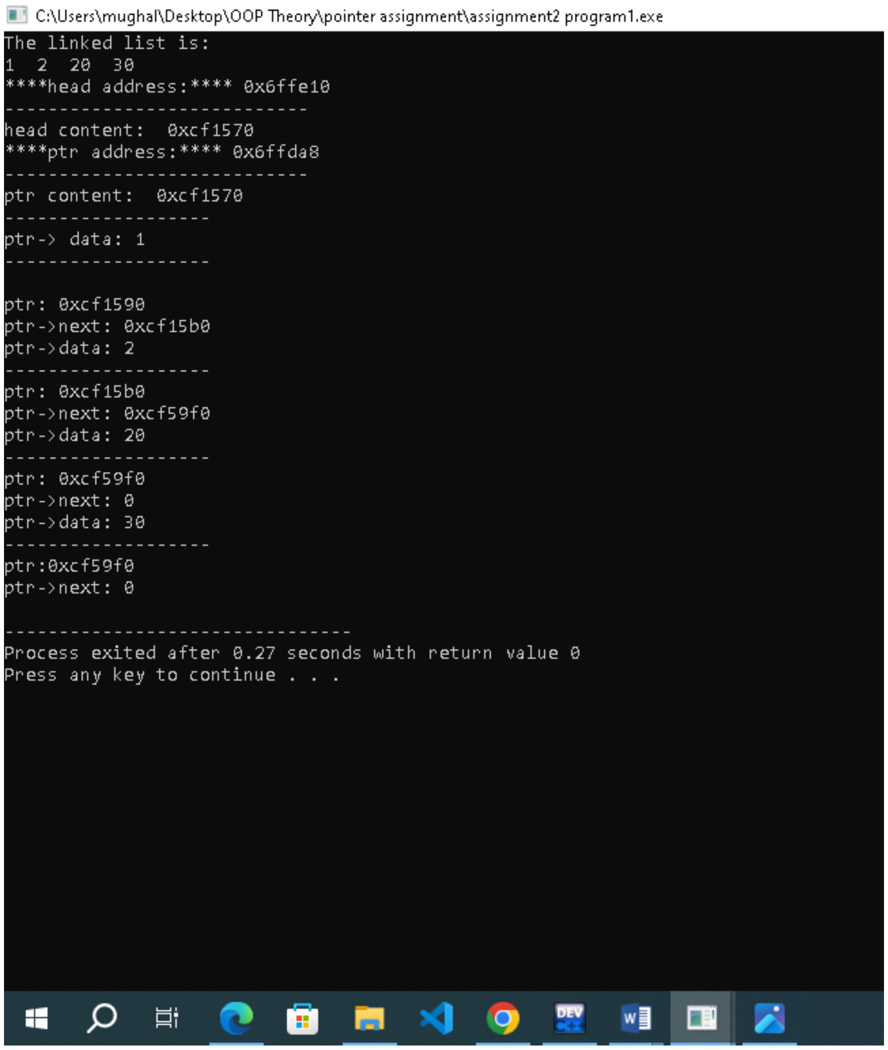
Task#01

Program:

```
#include <iostream>

using namespace std;

class Node{

private:

int data;

Node *next;

public:

Node * head;


Node(){

head==NULL;

}

void insert_at_end(int value){

Node *newnode= new Node();

if (head==NULL){

head= newnode;

head->data= value;

head->next= NULL;

}

else{

Node *ptr;

ptr= head;

while( ptr->next != NULL){

ptr= ptr->next;

}

ptr->next= newnode;
newnode->data= value;

newnode->next= NULL;


}


}


void display(){

cout<<"The linked list is:"<< endl;

if(head== NULL){

cout<<"Linked list is empty";

}

else{

Node *temp;

temp = head;

while( temp->next!=NULL){


cout<<temp->data<<"  ";

temp= temp->next;

}

cout<<temp->data<< endl;

}

}

void display1(){


Node *temp;

temp=head;

cout<<"****head address:**** "<< &head<< endl<<"----------------------------"<<endl<<"head content:  "<<
head<< endl;
```

```
cout<<"****ptr address:**** "<< &temp<< endl<<"---------------------------"<<endl<<"ptr content:  "<<
temp<< endl;

if(head==NULL){

cout<<"Linked list is empty";


}

else{

cout<<"-------------------"<<endl<<"ptr-> data: "<< temp->data<<endl<<"-------------------"<<endl<<endl;

while(temp->next!= NULL){

temp= temp->next;

cout<<"ptr: "<<temp<<endl<<"ptr->next: "<< temp->next<< endl<<"ptr->data: "<< temp-
>data<<endl<<"-------------------"<<endl;


}

cout<<"ptr:"<< temp<< endl<< "ptr->next: "<< temp->next<<endl;

}

}

};

int main(){

Node n;

n.insert_at_end(1);

n.insert_at_end(2);

n.insert_at_end(20);

n.insert_at_end(30);

n.display();

n.display1();

return 0;

}
```

Output:



C:\Users\mughal\Desktop\OOP Theory\pointer assignment\assignment2 program1.exe

```
The linked list is:
1  2  20  30
****head address:**** 0x6ffe10
----------------------------
head content:  0xcf1570
****ptr address:**** 0x6ffda8
----------------------------
ptr content:  0xcf1570
-------------------
ptr-> data: 1
-------------------

ptr: 0xcf1590
ptr->next: 0xcf15b0
ptr->data: 2
-------------------
ptr: 0xcf15b0
ptr->next: 0xcf59f0
ptr->data: 20
-------------------
ptr: 0xcf59f0
ptr->next: 0
ptr->data: 30
-------------------
ptr:0xcf59f0
ptr->next: 0

--------------------------------
Process exited after 0.27 seconds with return value 0
Press any key to continue . . .
```

Task 02

Program

```
#include <iostream>

using namespace std;

class single{

private:

int data;

single *next;

public:

single *head;

single(){

head=NULL;

}


void insert_at_begin_singly(int n){

single *newnode= new single();

if(head==NULL){


head= newnode;

head->data=n;

head->next=NULL;

}

else{

single *ptr;

ptr=newnode;

ptr->next=head;

ptr->data=n;

head=ptr;
```

```
}
display_singly();
}

void insert_at_end_singly(int n){
single *newnode= new single();
if(head==NULL){
head=newnode;
head->data=n;
head->next=NULL;
}
else{
single *ptr, *p;
ptr=head;
while(ptr->next!=NULL){
ptr=ptr->next;
}
p= newnode;
p->data=n;
p->next=NULL;
ptr->next=p;
}
display_singly();
}
void insertspecific_singly(int pos, int n){
single *newnode= new single();
if(head==NULL){
head = newnode;
head->data=n;
```

```
head->next=NULL;

}

else{

single *ptr;

ptr=head;

while(ptr->data!=pos){

ptr=ptr->next;

}

single *p;

p->data=n;

p->next=ptr->next;

ptr->next=p;

}

display_singly();


}


void del_begin_singly(){

single *ptr;

ptr= head;

if(head==NULL){

cout<<"No node to delete";

}

else{

head=ptr->next;

delete ptr;

ptr= NULL;
```

```
}

display_singly();

}

void del_end_singly(){

single *ptr, *ptr1;

ptr = head;

if(head==NULL){

cout<<"No node to delete";

}

else{

while(ptr->next !=NULL){

ptr1= ptr;

ptr= ptr->next;

}

ptr->next= NULL;

delete ptr;

ptr= NULL;

}

display_singly();

}


void delspecific_singly(int position){

if(head==NULL){

cout<<"No node to delete";

}

single *ptr, *ptr1;

while(ptr->data!= position){

ptr1= ptr;

ptr= ptr->next;
```

```
}
ptr1->next= ptr->next;
delete ptr;
ptr=NULL;
display_singly();
}

void display_singly(){
cout<<endl<<"Elements of linked list are= ";
if (head== NULL){
cout<<"Linked list is empty";
}
else{
single *ptr;
ptr=head;
while(ptr->next!=NULL){
ptr= ptr->next;
cout<<ptr->data<<"  ";
}
cout<<ptr->data;
}
cout<<endl<<endl;
}
};

class doubly{
private:
int data;
doubly* next;
```

```
doubly* prev;

public:

doubly *head;


doubly(){

head=NULL;

}


void insertbegin_doubly(int n){

doubly *newnode= new doubly();

doubly *ptr;

ptr=head;

if (head== NULL){

head= newnode;

head->data=n;

head->next=NULL;

head->prev= NULL;

}

else{

newnode->data= n;

newnode->next= ptr;

newnode->prev=NULL;

head= newnode;

}

display_doubly();

}

void insertend_doubly(int n){

doubly *newnode= new doubly();

doubly *ptr;
```

```
ptr=head;

if (head== NULL){

head= newnode;

head->data=n;

head->next=NULL;

head->prev= NULL;

}

else{

while(ptr->next!= NULL){

ptr= ptr->next;

}

ptr->next= newnode;

newnode->data=n;

newnode->next=NULL;

newnode->prev= ptr;


}

display_doubly();

}

void insertspecific_doubly(int n, int pos){

doubly *newnode= new doubly();

doubly *ptr, *ptr1;

while(ptr->data!= pos){

ptr= ptr->next;

ptr1= ptr;

}

newnode->data=n;

newnode->next= ptr;

ptr1->next= newnode;
```

```
newnode->prev=ptr1;

display_doubly();

}


void delbegin_doubly(){

if(head==NULL){

cout<<"Linked list is empty";


}

else{

doubly *ptr;

ptr= head;

head= ptr->next;

head->prev= ptr;

delete ptr;

ptr= NULL;

}

display_doubly();

}

void delend_doubly(){

doubly *ptr, *ptr1;

while(ptr!= NULL){

ptr1= ptr;

ptr=ptr->next;

}

ptr1->next= NULL;

delete ptr;

ptr= NULL;

display_doubly();
```

```
}

void delspecific_doubly(int pos){
doubly *ptr, *ptr1;
ptr= head;
while (ptr->data != pos){
ptr1= ptr;
ptr = ptr->next;
}
ptr1->next=ptr->next;
ptr->next->prev= ptr1;
delete ptr;
ptr= NULL;
display_doubly();

}

void display_doubly(){
doubly *ptr;
ptr= head;
if(head== NULL){
cout<<"Linked list is empty";
}
else{
cout<<"Elements of linked list are= ";
while(ptr!=NULL){
cout<< ptr->data;
ptr= ptr->next;
}
```

```
cout<<endl;

}



}

};


class circle{

private:

int data;

circle *next;

public:

circle *head;


circle(){

head= NULL;

}

void insert_endd(int n) {

if (head == NULL) {

head = new circle();

head->data = n;

head->next = head;  // Circular: Point to itself

} else {

circle* p, * ptr;

ptr = head;

while (ptr->next != head) {

ptr = ptr->next;

}
```

```
p = new circle();

p->data = n;

p->next = head;

ptr->next = p;

}

dispp();

}


void insert_begg(int n) {

if (head == NULL) {

head = new circle();

head->data = n;

head->next = head;  // Circular: Point to itself

} else {

circle* p, * ptr;

ptr = head;

while (ptr->next != head) {

ptr = ptr->next;

}


p = new circle();

p->data = n;

p->next = head;

ptr->next = p;

head=p;

}

dispp();

}
```

```
void insert_at_valuee(int pos, int n) {

if (head == NULL) {

cout << "List is empty. Cannot insert." << endl;

return;

}


circle* ptr;

ptr = head;

while (ptr->data != pos) {

ptr = ptr->next;

}


circle* p;

p = new circle();

p->data = n;

p->next = ptr->next;

ptr->next = p;

dispp();

}



void dispp() {

if (head == NULL) {

cout << "No data is in the list." << endl;

return;

}


circle* ptr = head;

do {
```

```
cout << ptr->data <<"\t";

ptr = ptr->next;

} while (ptr != head);

cout<<endl;

}



void del_begg() {

if (head == NULL) {

cout << "List is empty. Cannot delete." << endl;

return;

}


circle* temp = head;

circle* ptr = head;

while (ptr->next != head) {

ptr = ptr->next;

}

head = head->next;

ptr->next = head;

delete temp;

dispp();

}


void del_endd() {

if (head == NULL) {

cout << "List is empty. Cannot delete." << endl;

return;

}
```

```
if (head->next == head) {

delete head;

head = NULL;

return;

}


circle* ptr = head;

circle* prev = NULL;

while (ptr->next != head) {

prev = ptr;

ptr = ptr->next;

}


prev->next = head;

delete ptr;

dispp();

}


void del_at_valuee(int val) {

if (head == NULL) {

cout << "List is empty. Cannot delete." << endl;

return;

}


if (head->data == val) {

circle* temp = head;

circle* ptr = head;

while (ptr->next != head) {
```

```
ptr = ptr->next;

}

head = head->next;

ptr->next = head;

delete temp;

return;


}


circle* ptr = head;

circle* prev = NULL;

do {

prev = ptr;

ptr = ptr->next;

} while (ptr != head && ptr->data != val);


if (ptr != head) {

prev->next = ptr->next;

delete ptr;

} else {

cout << "Value not found in the list." << endl;

}

dispp();

}
```

```
};

int main(){

single obj1;

doubly obj2;

circle obj3;

int n, v, id, mn;

do

{

cout << "Select any One Linked List" << endl;

cout << "1: SINGLY" << endl;

cout << "2: DOUBLY" << endl;

cout << "3: CIRCULAR" << endl;

cin >> mn;

switch (mn){

case 1:

min:

cout << "Select any One Operation You want to Perform.." << endl;

cout << "1: INSERTION" << endl;

cout << "2: DELETION" << endl;

cin >> id;

switch (id)

{


case 1:

cout << "1: To add Node at Begining" << endl;

cout << "2: To add Node at End" << endl;

cout << "3: To add Node at Specific Location" << endl;

cout << "4: to Back" << endl;

cout << "5: to exit" << endl;
```

```
cin >> n;
switch (n)
{
case 1:
cout << "\nEnter the value to insert: ";
cin >> v;
obj1.insert_at_begin_singly(v);
break;
case 2:
cout << "\nEnter the value to insert: ";
cin >> v;
obj1.insert_at_end_singly(v);
break;
case 3:
int o, loc;
cout << "Enter location value: ";
cin >> loc;
cout << "Enter the value to insert: ";
cin >> v;
obj1.insertspecific_singly(loc, v);
break;
case 4:
goto min;
case 5:
exit(1);
default:
cout << "Choose valid Option" << endl;
break;
}
```

```
break;

system("pause");


case 2:


cout << "1: To Delete Node from Begining" << endl;

cout << "2: To Delete Node from End" << endl;

cout << "3: To Delete Specific Node" << endl;

cout << "4: to Back" << endl;

cout << "5: to exit" << endl;

cin >> n;

switch (n)

{

case 1:

cout<<"Node deleted from Begining....";

obj1.del_begin_singly();

break;

case 2:

cout<<"Node deleted from END....";

obj1.del_end_singly();

break;

}

case 3:

cout << "Enter the node value to Delete: ";

cin >> v;

obj1.delspecific_singly(v);

obj1.display_singly();

break;

default:
```

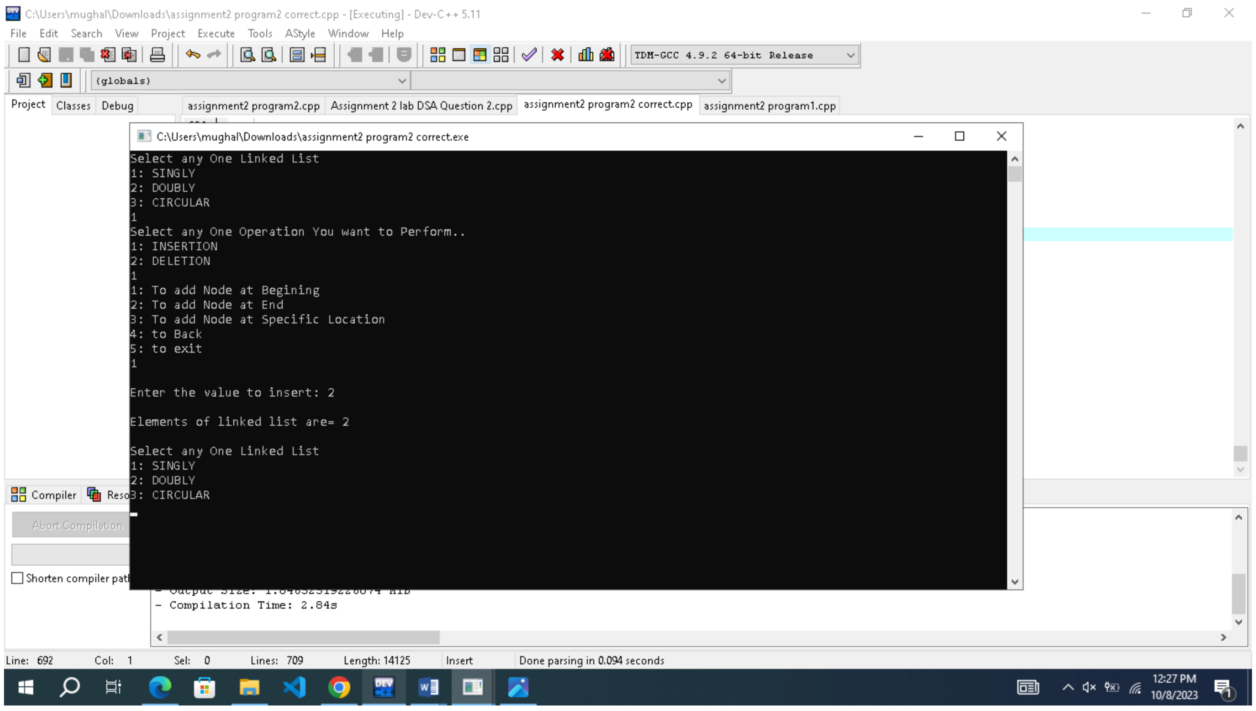
```
cout << "Choose valid Option" << endl;

break;


}

break;

//end of case singly

//----------------------------------------------

case 2:

tg:

cout << "Select any One Operation You want to Perform.." << endl;

cout << "1: INSERTION" << endl;

cout << "2: DELETION" << endl;

cin >> id;

switch (id){

//insertion in doubly

case 1:


cout << "1: To add Node at Begining" << endl;

cout << "2: To add Node at End" << endl;

cout << "3: To add Node at Specific Location" << endl;

cout << "4: to Back" << endl;

cout << "5: to exit" << endl;

cin >> n;

switch (n)

{

case 1:

cout << "\nEnter the value to insert: ";

cin >> v;

obj2.insertbegin_doubly(v);
break;

case 2:
cout << "\nEnter the value to insert: ";
cin >> v;
obj2.insertend_doubly(v);
break;

case 3:
int o, loc;
cout << "Enter location value: ";
cin >> loc;
cout << "Enter the value to insert: ";
cin >> v;
obj2.insertspecific_doubly(v,loc);

case 4:
goto tg;
case 5:
exit(1);
default:
cout << "Choose valid Option" << endl;
break;
}
break;
//deletion in doubly
case 2:
cout << "1: To Delete Node from Begining" << endl;
cout << "2: To Delete Node from End" << endl;
cout << "3: To Delete Specific Node" << endl;
cout << "4: to Back" << endl;
cout << "5: to exit" << endl;
cin >> n;
switch (n)
{
case 1:
cout<<"Node deleted from Begining....";
obj2.delbegin_doubly();
break;
case 2:
cout<<"Node deleted from END....";
obj2.delend_doubly();
break;
case 3:
cout << "Enter the node value to Delete: ";
cin >> v;
obj2.delspecific_doubly(v);

break;
default:
cout << "Choose valid Option" << endl;
break;
}


}
break;
//end of case doubly
```

```
//-----------------------------------
case 3:
gb:
cout << "Select any One Operation You want to Perform.." << endl;
cout << "1: INSERTION" << endl;
cout << "2: DELETION" << endl;
cin >> id;
switch (id){
//insertion in Circular
case 1:

cout << "1: To add Node at Begining" << endl;
cout << "2: To add Node at End" << endl;
cout << "3: To add Node at Specific Location" << endl;
cout << "4: to Back" << endl;
cout << "5: to exit" << endl;
cin >> n;
switch (n)
{
case 1:
cout << "\nEnter the value to insert: ";
cin >> v;
obj3.insert_begg(v);
break;

case 2:
cout << "\nEnter the value to insert: ";
cin >> v;
obj3.insert_endd(v);
break;

case 3:
int o, loc;
cout << "Enter location value: ";
cin >> loc;
cout << "Enter the value to insert: ";
cin >> v;
obj3.insert_at_valuee(loc, v);

case 4:
goto gb;
case 5:
exit(1);
default:
cout << "Choose valid Option" << endl;
break;
}
break;
//deletion in Circular
case 3:
obj3.dispp();
case 2:
cout << "1: To Delete Node from Begining" << endl;
cout << "2: To Delete Node from End" << endl;
cout << "3: To Delete Specific Node" << endl;
cout << "4: to Back" << endl;
cout << "5: to exit" << endl;
cin >> n;
```

```
switch (n)

{

case 1:

cout<<"Node deleted from Begining....";

obj3.del_begg();

break;

case 2:

cout<<"Node deleted from END....";

obj3.del_endd();

break;

case 3:

cout << "Enter the node value to Delete: ";

cin >> v;

obj3.del_at_valuee(v);

break;

default:

cout << "Choose valid Option" << endl;

break;

}



}

break;

//end of case circular





default:
```

```
cout << "Choose valid Option" << endl;

break;


//end of singly/doubly switch

}




} while (n != 4);

system("pause");

return 0;

}
```

Output:



C:\Users\mughal\Downloads\assignment2 program2 correct.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
assignment2 program2.cpp
Assignment 2 lab DSA Question 2.cpp
assignment2 program2 correct.cpp
assignment2 program1.cpp
C:\Users\mughal\Downloads\assignment2 program2 correct.exe
Compiler
Abort Compilation
Shorten compiler paths
Line: 692 Col: 1 Sel: 0 Lines: 709 Length: 14125 Insert Done parsing in 0.094 seconds
12:27 PM
10/8/2023

```
Select any One Linked List
1: SINGLY
2: DOUBLY
3: CIRCULAR
1
Select any One Operation You want to Perform..
1: INSERTION
2: DELETION
1
1: To add Node at Begining
2: To add Node at End
3: To add Node at Specific Location
4: to Back
5: to exit
1

Enter the value to insert: 2

Elements of linked list are= 2

Select any One Linked List
1: SINGLY
2: DOUBLY
3: CIRCULAR
```

```
- Compilation Time: 2.84s
```