ASSIGNMENT # 01

Submitted to : Mam Yasmeen jana

Submitted ty : Ammar Shafique

Roll no : FA21-BCS-008

Subject : DSA Lab

TASK 1:

```
void Display_SLL(){

        if(head==NULL){
                cout<<"\n Linked-List is Empty!. ";
        }
        else{
        node *temp;
        temp=head;
        cout<<" ==> Linked List { ";
        while(temp!=NULL){

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

        }
        cout<<" }";
}

}//display function
////////////////////////////////////////////////////////////////////////////
void Display_SLL_ADRS(){

        if(head==NULL){
                cout<<"\n Linked-List is Empty!. ";
        }
        else{
        node *temp;
```

```
temp=head;
system("cls");
cout<<"\n ==> Linked List { ";
while(temp!=NULL){

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

}
cout<<" }\n\n";
//.....

cout<<"\n        >> HEAD << \n";
cout<<"\n  ADDRESS = "<<&head;
cout<<"\n  CONTENT = "<<head;
cout<<"\n";
int i=1;
temp=head;
while(temp!=NULL){
        cout<<"\n        >> NODE "<<i<<" <<\n\n";
        cout<<"  ADDRESS = "<<temp;
        cout<<"\n  NEXT    = "<<temp->next;

        cout<<"\n  DATA    = "<<temp->data<<" \n";
        i++;
        temp=temp->next;

}
```

}

}//display sll with addresses

```
////////////////////////////////////////////////////////////////////////////
void Display_SLL_ADRS(){

    if(head==NULL){
        cout<<"\n Linked-List is Empty!. ";
    }
    else{
    node *temp;
    temp=head;
    system("cls");
    cout<<"\n ==> Linked List { ";
    while(temp!=NULL){

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

    }
    cout<<" }\n\n";
    //..............................

    cout<<"\n          >> HEAD << \n";
    cout<<"\n  ADDRESS = "<<&head;
    cout<<"\n  CONTENT = "<<head;
    cout<<"\n";
    int i=1;
    temp=head;
    while(temp!=NULL){
        cout<<"\n          >> NODE "<<i<<" <<\n\n";
        cout<<"  ADDRESS = "<<temp;
        cout<<"\n  NEXT    = "<<temp->next;

        cout<<"\n  DATA    = "<<temp->data<<" \n";
        i++;
        temp=temp->next;

    }


}//display sll with adresses
////////////////////////////////////////////////////////////////////////////
void Del_end_SLL(){
```

```
C:\Users\Ammar\Desktop\Linked list.exe

         >> HEAD <<

ADDRESS = 0x79fdf8
CONTENT = 0x1e1430

         >> NODE 1 <<

ADDRESS = 0x1e1430
NEXT    = 0x1e5780
DATA    = 1

         >> NODE 2 <<

ADDRESS = 0x1e5780
NEXT    = 0x1e57b0
DATA    = 2

         >> NODE 3 <<

ADDRESS = 0x1e57b0
NEXT    = 0x1e57e0
DATA    = 3

         >> NODE 4 <<

ADDRESS = 0x1e57e0
NEXT    = 0
DATA    = 4
```

void Display_CLL_ADRS(){ // CLL= Circular Linked-List

```
if(head==NULL){
        cout<<"\n Linked-List is Empty!. ";
}
else{
node *temp;
temp=head;
system("cls");
cout<<"\n ==> Linked List { ";
while(temp->next!=head){
```

```
        cout<<temp->data<<" ";
        temp=temp->next;

}
cout<<" }\n\n";
//.............................

cout<<"\n        >> HEAD << \n";
cout<<"\n  ADDRESS = "<<&head;
cout<<"\n  CONTENT = "<<head;
cout<<"\n";
int i=1;
temp=head;
while(temp->next!=head){
        cout<<"\n        >> NODE "<<i<<" <<\n\n";
        cout<<"  ADDRESS = "<<temp;
        cout<<"\n  NEXT    = "<<temp->next;

        cout<<"\n  DATA    = "<<temp->data<<" \n";
        i++;
        temp=temp->next;

}
        cout<<"\n        >> NODE "<<i<<" <<\n\n";
        cout<<"  ADDRESS = "<<temp;
        cout<<"\n  NEXT    = "<<temp->next;

        cout<<"\n  DATA    = "<<temp->data<<" \n";
```

}

}//display Cll with addresses

```
void Display_CLL_ADRS(){ // CLL= Circular Linked-List

    if(head==NULL){
        cout<<"\n Linked-List is Empty!. ";
    }
    else{
    node *temp;
    temp=head;
    system("cls");
    cout<<"\n ==> Linked List { ";
    while(temp->next!=head){

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

    }
    cout<<" }\n\n";
    //..............................

    cout<<"\n          >> HEAD << \n";
    cout<<"\n  ADDRESS = "<<&head;
    cout<<"\n  CONTENT = "<<head;
    cout<<"\n";
    int i=1;
    temp=head;
    while(temp->next!=head){
        cout<<"\n          >> NODE "<<i<<" <<\n\n";
        cout<<"  ADDRESS = "<<temp;
        cout<<"\n  NEXT    = "<<temp->next;

        cout<<"\n  DATA    = "<<temp->data<<" \n";
        i++;
        temp=temp->next;

    }
        cout<<"\n          >> NODE "<<i<<" <<\n\n";
        cout<<"  ADDRESS = "<<temp;
        cout<<"\n  NEXT    = "<<temp->next;

        cout<<"\n  DATA    = "<<temp->data<<" \n";

}//display Cll with adresses
 ////////////////////////////////////////////////////////////
```

```
C:\Users\Ammar\Desktop\Linked list.exe

         >> HEAD <<

ADDRESS = 0x79fdd8
CONTENT = 0x1e5810

         >> NODE 1 <<

ADDRESS = 0x1e5810
NEXT    = 0x1e5840
DATA    = 1

         >> NODE 2 <<

ADDRESS = 0x1e5840
NEXT    = 0x1e5870
DATA    = 2

         >> NODE 3 <<

ADDRESS = 0x1e5870
NEXT    = 0x1e58a0
DATA    = 3

         >> NODE 4 <<

ADDRESS = 0x1e58a0
NEXT    = 0x1e5810
DATA    = 4
```

void Display_DLL_ADRS(){

if(head==NULL){

cout<<"\n Linked-List is Empty!. ";

}

else{

node *temp;

temp=head;

system("cls");

```
        cout<<"\n ==> Linked List { ";
        while(temp!=NULL){

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

        }
        cout<<" }\n\n";
        //.............................

        cout<<"\n        >> HEAD << \n";
        cout<<"\n  ADDRESS = "<<&head;
        cout<<"\n  CONTENT = "<<head;
        cout<<"\n";
        int i=1;
        temp=head;
        while(temp!=NULL){
                cout<<"\n        >> NODE "<<i<<" <<\n\n";
                cout<<"  ADDRESS = "<<temp;
                cout<<"\n  NEXT    = "<<temp->next;

                cout<<"\n  DATA    = "<<temp->data<<" \n";
                i++;
                temp=temp->next;

        }

}
    }//display DLL with addresses
```

```
////////////////////////////////////////////////////////////////////////////
void Display_DLL_ADRS(){

    if(head==NULL){
        cout<<"\n Linked-List is Empty!. ";
    }
    else{
    node *temp;
    temp=head;
    system("cls");
    cout<<"\n ==> Linked List { ";
    while(temp!=NULL){

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

    }
    cout<<" }\n\n";
    //..............................

    cout<<"\n          >> HEAD << \n";
    cout<<"\n  ADDRESS = "<<&head;
    cout<<"\n  CONTENT = "<<head;
    cout<<"\n";
    int i=1;
    temp=head;
    while(temp!=NULL){
        cout<<"\n          >> NODE "<<i<<" <<\n\n";
        cout<<"  ADDRESS = "<<temp;
        cout<<"\n  NEXT    = "<<temp->next;

        cout<<"\n  DATA    = "<<temp->data<<" \n";
        i++;
        temp=temp->next;

    }


}//display DLL with adresses
////////////////////////////////////////////////////////////
void Del_end_DLL(){
```

```
C:\Users\Ammar\Desktop\Linked list.exe

         >> HEAD <<

ADDRESS = 0x79fdb8
CONTENT = 0x1e58d0

         >> NODE 1 <<

ADDRESS = 0x1e58d0
NEXT    = 0x1e5900
DATA    = 1

         >> NODE 2 <<

ADDRESS = 0x1e5900
NEXT    = 0x1e5930
DATA    = 2

         >> NODE 3 <<

ADDRESS = 0x1e5930
NEXT    = 0x1e5960
DATA    = 3

         >> NODE 4 <<

ADDRESS = 0x1e5960
NEXT    = 0
DATA    = 4
```

TASK 2:

```
int cw=0;
int dw=0;
int m=0;

do{
    system("cls");
cout<<"<''''''' MAIN-MENU '''''''>\n";
cout<<" 1. Singular Linked-List.\n";
cout<<" 2. Circular Linked-List.\n";
cout<<" 3. Doubly Linked-List.\n";
cout<<"==> Your Choice is ";
int m;
cin>>m;

switch(m){

case 1:{ ////////////////////////////////////////////////  SLL
    int sw=0;
    do{
        system("cls");
cout<<"<''''''' Singly Linked-List MENU '''''''>\n";
cout<<" 1. Insert At beginning.\n";
cout<<" 2. Insert At END.\n";
cout<<" 3. Insert At Specific Location.\n";
cout<<" 4. Delete At Beginning.\n";
cout<<" 5. Delete At END.\n";
cout<<" 6. Delete At Specific Location.\n";
cout<<" 7. Reverse Linked-List.\n";
cout<<" 8. DISPLAY ( ADDRESS ).\n";
cout<<" 9. DISPLAY ( SIMPLE ). \n";
cout<<" 0. To go Back. \n\n";

cout<<"==> Your Choice is ";
int choice;
cin>>choice;


switch(choice){
    case 1:{
        cout<<"\n Enter Value ";
        int v;
        cin>>v;
        n1.Insert_Begin_SLL(v);
        break;
        }
        case 2:{
case 2:{////////////////////////////////////////////////  CLL  ////////////////////////////////////////////////////////


    do{
        system("cls");
cout<<"<''''''' Circular Linked-List MENU '''''''>\n";
cout<<" 1. Insert At beginning.\n";
cout<<" 2. Insert At END.\n";
cout<<" 3. Insert At Specific Location.\n";
cout<<" 4. Delete At Beginning.\n";
cout<<" 5. Delete At END.\n";
cout<<" 6. Delete At Specific Location.\n";
cout<<" 7. Reverse Linked-List.\n";
cout<<" 8. DISPLAY ( ADDRESS ).\n";
cout<<" 9. DISPLAY ( SIMPLE ). \n";
cout<<" 0. To go Back. \n\n";

cout<<"==> Your Choice is ";
int choice;
cin>>choice;
switch(choice){
    case 1:{
        cout<<"\n Enter Value ";
        int v;
        cin>>v;
        n2.Insert_Begin_CLL(v);
        break;
        }
        case 2:{
            cout<<"\n Enter Value ";
        int v;
        cin>>v;
        n2.Insert_END_CLL(v);
            break;
```

C:\Users\Ammar\Desktop\Linked list.exe

```
<''''''' Singly Linked-List MENU '''''''>
 1. Insert At beginning.
 2. Insert At END.
 3. Insert At Specific Location.
 4. Delete At Beginning.
 5. Delete At END.
 6. Delete At Specific Location.
 7. Reverse Linked-List.
 8. DISPLAY ( ADDRESS ).
 9. DISPLAY ( SIMPLE ).
 0. To go Back.

==> Your Choice is
```

C:\Users\Ammar\Desktop\Linked list.exe

```
<''''''' Circular Linked-List MENU '''''''>
 1. Insert At beginning.
 2. Insert At END.
 3. Insert At Specific Location.
 4. Delete At Beginning.
 5. Delete At END.
 6. Delete At Specific Location.
 7. Reverse Linked-List.
 8. DISPLAY ( ADDRESS ).
 9. DISPLAY ( SIMPLE ).
 0. To go Back.

==> Your Choice is
```

```
case 3:{////////////////////////////////////////////// DLL ////////////////////////////////////////////////////
    do{
        system("cls");
cout<<"<''''''' Doubly Linked-List MENU '''''''>\n";
cout<<" 1. Insert At beginning.\n";
cout<<" 2. Insert At END.\n";
cout<<" 3. Insert At Specific Location.\n";
cout<<" 4. Delete At Beginning.\n";
cout<<" 5. Delete At END.\n";
cout<<" 6. Delete At Specific Location.\n";
cout<<" 7. Reverse Linked-List.\n";
cout<<" 8. DISPLAY ( ADDRESS ).\n";
cout<<" 9. DISPLAY ( SIMPLE ). \n";
cout<<" 0. To go Back. \n\n";

cout<<"==> Your Choice is ";
int choice;
cin>>choice;


switch(choice){
    case 1:{
        cout<<"\n Enter Value ";
        int v;
        cin>>v;
        n3.Insert_Begin_DLL(v);
        break;
        }
        case 2:{
            cout<<"\n Enter Value ";
        int v;
        cin>>v;
        n3.Insert_END_DLL(v);
```

```
C:\Users\Ammar\Desktop\Linked list.exe
<''''''' Doubly Linked-List MENU '''''''>
 1. Insert At beginning.
 2. Insert At END.
 3. Insert At Specific Location.
 4. Delete At Beginning.
 5. Delete At END.
 6. Delete At Specific Location.
 7. Reverse Linked-List.
 8. DISPLAY ( ADDRESS ).
 9. DISPLAY ( SIMPLE ).
 0. To go Back.

==> Your Choice is
```

```
#include <iostream>
#include <windows.h>
#include <conio.h>
using namespace std;
        const char* title = "Notification";
  const char* message = " Invalid Choice! ";
  const char* message_del = " Value Deleted Successfully!.";
class node{
        private:
                int data;
                node *prev;
                node *next;
        public:
                node *head;
                node(){
                        head=NULL;
                }
                        void Insert_END_SLL(int n){
```

```
        if(head==NULL){
                node *new_node=new node();
                new_node->data=n;
                new_node->next=NULL;
                head=new_node;
        }
        else {
        node *new_node=new node();
        node *temp;
        temp=head;
        while(temp->next!=NULL){
                temp=temp->next;
        }
        new_node->data=n;
        new_node->next=NULL;
        temp->next=new_node;

}

        } // insert END function
        ////////////////////////////////////////////////////////////////////////////

        void Insert_Begin_SLL(int n){
                        if(head==NULL){
                node *new_node=new node();
                new_node->data=n;
                new_node->next=NULL;
                head=new_node;
```

```
        }
        else {
        node *new_node=new node();
        node *temp;
        temp=head;
        new_node->data=n;
        new_node->next=temp;
        head=new_node;



        }  //else
}//insert Begin function
////////////////////////////////////////////////////////////////////////////
        void Display_SLL(){

                if(head==NULL){
                        cout<<"\n Linked-List is Empty!. ";
                }
                else{
                node *temp;
                temp=head;
                cout<<" ==> Linked List { ";
                while(temp!=NULL){

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

                }
                cout<<" }";
```

```
}

}//display function
////////////////////////////////////////////////////////////////////////////
void Display_SLL_ADRS(){

        if(head==NULL){
                cout<<"\n Linked-List is Empty!. ";
        }
        else{
        node *temp;
        temp=head;
        system("cls");
        cout<<"\n ==> Linked List { ";
        while(temp!=NULL){

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

        }
        cout<<" }\n\n";
        //.....................................

        cout<<"\n         >> HEAD << \n";
        cout<<"\n  ADDRESS = "<<&head;
        cout<<"\n  CONTENT = "<<head;
        cout<<"\n";
        int i=1;
        temp=head;
```

```
		while(temp!=NULL){
			cout<<"\n	>> NODE "<<i<<" <<\n\n";
			cout<<"  ADDRESS = "<<temp;
			cout<<"\n  NEXT    = "<<temp->next;

			cout<<"\n  DATA    = "<<temp->data<<" \n";
			i++;
			temp=temp->next;

		}

	}
		}//display sll with adresses
		////////////////////////////////////////////////////////////////////////
		void Del_end_SLL(){
			if(head==NULL){
			message_del="\n Linked-List is Empty Already!!!";
					MessageBox(NULL, message_del, title,
MB_ICONINFORMATION | MB_OK);

			}
			else{

			node *temp1,*temp2;
			temp1=head;
			while(temp1->next!=NULL){
				temp2=temp1;
				temp1=temp1->next;
			}
```

```
                temp2->next=NULL;
                delete temp1;
                MessageBox(NULL, message_del, title, MB_ICONINFORMATION |
MB_OK);
        }
        }
    ////////////////////////////////////////////////////////////////////////
    void Insert_spec_SLL(int v,int sp){
        node *new_node= new node();
        node *temp;
        new_node->data=v;
        temp=head;
        while(temp->data!=sp){
            temp=temp->next;
        }
        new_node->next=temp->next;
        temp->next=new_node;

    }// insert SLL specific location

    ////////////////////////////////////////////////////////////////////////
    void Del_BEGIN_SLL(){
        if(head==NULL){
            cout<<"\n\n ...>> There's NO Node !!! <<...";
        }
        else if(head->next==NULL){
            head=NULL;
            cout<<"\n ==> DELETED SUCCESSFULLY !!! <==\n";
            MessageBox(NULL, message_del, title, MB_ICONINFORMATION |
MB_OK);
```

```
		}
		else
		{
			head=head->next;
			cout<<"\n ==> DELETED SUCCESSFULLY !!! <==\n";
			MessageBox(NULL, message_del, title, MB_ICONINFORMATION |
MB_OK);

		}

	}
	////////////////////////////////////////////////////////////////////////////
	void Del_Spec_SLL(int sp){
		if (head->data==sp){
			head=head->next;
		}
		else{

		node *temp1,*temp2;
		temp1=head;
		while(temp1->data!=sp){
			temp2=temp1;
			temp1=temp1->next;
		}
		temp2->next=temp1->next;}

	}
	////////////////////////////////////////////////////////////////////////////
	void Reverse_SLL(){
```

```
        node *temp1,*temp2,*temp3;
        temp1=head;
        temp2=temp1->next;
        temp1->next=NULL;
        temp3=temp2;
        while(temp3->next!=NULL){
                temp3=temp2->next;
                temp2->next=temp1;
                temp1=temp2;
                temp2=temp3;
        }
        temp2->next=temp1;
        head=temp3;
    }

    ////////////////////////////////////////////////////////////////////////
    ////////////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////////////////CLL/////////////////////////////////////////////
        void Insert_END_CLL(int n){

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

```
        node *temp;
        temp=head;
        while(temp->next!=head){
                temp=temp->next;
        }
        new_node->data=n;
        temp->next=new_node;
        new_node->next=head;

}

        }  // insert END function
////////////////////////////////////////////////////
        void Insert_Begin_CLL(int n){
                        if(head==NULL){
                node *new_node=new node();
                new_node->data=n;
                head=new_node;
                new_node->next=head;
        }
        else {
        node *new_node=new node();
        node *temp;
        temp=head;
        new_node->next=temp;

        while(temp->next!=head){
                temp=temp->next;
        }
```

```
            new_node->data=n;
        temp->next=new_node;
        head=new_node;



            }  //else
        }//insert Begin function
////////////////////////////////////////////////////////////////////
            void Display_CLL(){
                if(head==NULL){
                    cout<<"\n Linked-List is Empty!. ";
                }
                else{

                node *temp;
                temp=head;
                cout<<" ==> Linked List { ";
                while(temp->next!=head){

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

                }
                cout<<temp->data<<" ";
                cout<<" }";
            }//else display
            }//display function
        ////////////////////////////////////////////////////////////////////
```

```
void insert_spec_CLL(int sp,int value){//sp=specific position
node *new_node= new node();
node *temp;

temp=head;
while(temp->data!=sp){
        temp=temp->next;
}
new_node->data=value;
new_node->next=temp->next;
temp->next=new_node;

}
////////////////////////////////////////////////////////////////////////

void Display_CLL_ADRS(){ // CLL= Circular Linked-List

        if(head==NULL){
                cout<<"\n Linked-List is Empty!. ";
        }
        else{
        node *temp;
        temp=head;
        system("cls");
        cout<<"\n ==> Linked List { ";
        while(temp->next!=head){

                cout<<temp->data<<" ";
                temp=temp->next;
```

```
        }
        cout<<" }\n\n";
        //.............................

        cout<<"\n        >> HEAD << \n";
        cout<<"\n  ADDRESS = "<<&head;
        cout<<"\n  CONTENT = "<<head;
        cout<<"\n";
        int i=1;
        temp=head;
        while(temp->next!=head){
                cout<<"\n        >> NODE "<<i<<" <<\n\n";
                cout<<"  ADDRESS = "<<temp;
                cout<<"\n  NEXT    = "<<temp->next;

                cout<<"\n  DATA    = "<<temp->data<<" \n";
                i++;
                temp=temp->next;

        }
                cout<<"\n        >> NODE "<<i<<" <<\n\n";
                cout<<"  ADDRESS = "<<temp;
                cout<<"\n  NEXT    = "<<temp->next;

                cout<<"\n  DATA    = "<<temp->data<<" \n";

}
        }//display Cll with adresses
```

```
////////////////////////////////////////////////////////////////////////
void Del_END_CLL(){
	if(head->next==head){
		head = NULL;
	}
	else{
	node *temp1,*temp2;
	temp1=head;
	while(temp1->next!=head){
		temp2=temp1;
		temp1=temp1->next;
		}
		temp2->next=head;
	}
}

////////////////////////////////////////////////////////////////////////
void Reverse_CLL(){
	node *temp1,*temp2,*temp3,*temp4;
	temp1=head;
	temp2=temp1->next;
	temp3=temp2;
	temp4=head;
	while(temp3->next!=head){
		temp3=temp2->next;
		temp2->next=temp1;
		temp1=temp2;
		temp2=temp3;
	}
```

```
        temp2->next=temp1;
        head=temp2;
        temp4->next=head;

}
//////////////////////////////////////////////////////////////////////////
void Del_Spec_CLL(int sp){
        if(head->data==sp){
                Del_Begin_CLL();
        }
        else{
                node *temp1,*temp2;
                temp1=head;
                while(temp1->data!=sp){
                        temp2=temp1;
                        temp1=temp1->next;

                }
                temp2->next=temp1->next;
                delete temp1;

        }

}
//////////////////////////////////////////////////////////////////////////
void Del_Begin_CLL(){
        node *temp;
        temp=head;
        while(temp->next!=head){
```

```
                temp=temp->next;
        }
        temp->next=head->next;
        head=head->next;
    }

    ////////////////////////////////////////////////////////////////////////
    //  UP  ////////////////////// CLL -////////////////////////////////////  UP
        void Insert_END_DLL(int n){
        node *new_node=new node();
                new_node->data=n;

    if(head==NULL){
                new_node->next=NULL;
        new_node->prev=new_node;
        head=new_node;
    }
    else {
    node *temp;
    temp=head;
    while(temp->next!=NULL){
        temp=temp->next;
    }
    new_node->next=NULL;
    temp->next=new_node;
    new_node->prev=temp;

}
```

```
	} // insert END function
	////////////////////////////////////////////////////////////////////////////

	void Insert_Begin_DLL(int n){
			if(head==NULL){
		node *new_node=new node();
		new_node->data=n;
		new_node->next=NULL;
		new_node->prev=new_node;
		head=new_node;
	}
	else {
	node *new_node=new node();
	node *temp;
	temp=head;
	temp->prev=new_node;
	new_node->data=n;
	new_node->next=head;
	head=new_node;



	} //else
}//insert Begin function
////////////////////////////////////////////////////////////////////////////
	void Display_DLL(){

		if(head==NULL){
			cout<<"\n Linked-List is Empty!. ";
		}
```

```
        else{
        node *temp;
        temp=head;
        cout<<" ==> Linked List { ";
        while(temp!=NULL){

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

        }
        cout<<" }";
}

}//display function
////////////////////////////////////////////////////////////////////////////
void Display_DLL_ADRS(){

        if(head==NULL){
                cout<<"\n Linked-List is Empty!. ";
        }
        else{
        node *temp;
        temp=head;
        system("cls");
        cout<<"\n ==> Linked List { ";
        while(temp!=NULL){

                cout<<temp->data<<" ";
                temp=temp->next;
```

```
            }
            cout<<" }\n\n";
            //.....................................

            cout<<"\n         >> HEAD << \n";
            cout<<"\n  ADDRESS = "<<&head;
            cout<<"\n  CONTENT = "<<head;
            cout<<"\n";
            int i=1;
            temp=head;
            while(temp!=NULL){
                  cout<<"\n         >> NODE "<<i<<" <<\n\n";
                  cout<<"  ADDRESS = "<<temp;
                  cout<<"\n  NEXT    = "<<temp->next;

                  cout<<"\n  DATA    = "<<temp->data<<" \n";
                  i++;
                  temp=temp->next;

            }

}
      }//display DLL with adresses
      ////////////////////////////////////////////////////////////////////////////
      void Del_end_DLL(){
            if(head==NULL){
            message_del="\n Linked-List is Empty Already!!!";
            getch();
```

```
                }
                else if(head->next==NULL){
                        head=NULL;
                        MessageBox(NULL, message_del, title, MB_ICONINFORMATION |
MB_OK);

                }
                else{

                node *temp1,*temp2;
                temp1=head;
                while(temp1->next!=NULL){
                        temp2=temp1;
                        temp1=temp1->next;
                }
                temp2->next=NULL;
                delete temp1;
                MessageBox(NULL, message_del, title, MB_ICONINFORMATION |
MB_OK);
        }
        }
    ////////////////////////////////////////////////////////////////////////
    void Insert_spec_DLL(int v,int sp){
        node *new_node= new node();
        node *temp;
        new_node->data=v;
        temp=head;
        while(temp->data!=sp){
                temp=temp->next;
        }
```

```
            new_node->next=temp->next;
            new_node->prev=temp;
            temp->next=new_node;

        }// insert SLL specific location

        ////////////////////////////////////////////////////////////////////////
        void Del_BEGIN_DLL(){
            if(head==NULL){
                cout<<"\n\n ...>> There's NO Node !!! <<...";
            }
            else if(head->next==NULL){
                head=NULL;
                cout<<"\n ==> DELETED SUCCESSFULLY !!! <==\n";
                MessageBox(NULL, message_del, title, MB_ICONINFORMATION |
MB_OK);
            }
            else
            {
                head=head->next;
                cout<<"\n ==> DELETED SUCCESSFULLY !!! <==\n";
                MessageBox(NULL, message_del, title, MB_ICONINFORMATION |
MB_OK);

            }

        }
        ////////////////////////////////////////////////////////////////////////
        void Del_Spec_DLL(int sp){
            if (head->data==sp){
```

```
            head=head->next;
            head->prev=head;
        }
        else{

        node *temp1,*temp2,*temp3;
        temp1=head;
        while(temp1->data!=sp){
            temp1=temp1->next;
        }
        temp2=temp1->prev;
        temp3=temp1->next;
        temp2->next=temp3;
        temp3->prev=temp2;
        }

}
////////////////////////////////////////////////////////////////////////////
void Reverse_DLL(){
        node *temp1,*temp2,*temp3,*temp4;
        temp1=head;
        temp2=temp1;
        temp4=temp1;
        while(temp1!=NULL){
            temp3=temp1->prev;
            temp1->prev=temp1->next;
            temp1->next=temp3;
            temp2=temp1;
            temp1=temp1->prev;
```

```
            }
            temp4->next=NULL;
            temp2->prev=temp2;
            head=temp2;
        }

        ////////////////////////////////////////////////////////////////////////
            ////////////////////////////////////////////////////////////////////////
};
int main(){
node n1;
node n2;
node n3;
int sw=0;
int cw=0;
int dw=0;
int m=0;

do{
        system("cls");
cout<<"<'''''' MAIN-MENU ''''''>\n";
cout<<" 1. Singular Linked-List.\n";
cout<<" 2. Circular Linked-List.\n";
cout<<" 3. Doubly Linked-List.\n";
cout<<"==> Your Choice is ";
int m;
cin>>m;

switch(m){
```

```
case 1:{ ////////////////////////////////////////////// SLL
//////////////////////////////////////////////////////
        int sw=0;
        do{
                system("cls");
cout<<"<''''''' Singly Linked-List MENU '''''''>\n";
cout<<" 1. Insert At beginning.\n";
cout<<" 2. Insert At END.\n";
cout<<" 3. Insert At Specific Location.\n";
cout<<" 4. Delete At Beginning.\n";
cout<<" 5. Delete At END.\n";
cout<<" 6. Delete At Specific Location.\n";
cout<<" 7. Reverse Linked-List.\n";
cout<<" 8. DISPLAY ( ADDRESS ).\n";
cout<<" 9. DISPLAY ( SIMPLE ). \n";
cout<<" 0. To go Back. \n\n";

cout<<"==> Your Choice is ";
int choice;
cin>>choice;


switch(choice){
        case 1:{
                cout<<"\n Enter Value ";
                int v;
                cin>>v;
                n1.Insert_Begin_SLL(v);
```

```
break;
}
case 2:{
        cout<<"\n Enter Value ";
int v;
cin>>v;
n1.Insert_END_SLL(v);

        break;
}
case 3:{
        cout<<"\n Enter Value ";
        int v;
cin>>v;
cout<<"\n Enter location, you want to put after that value ";
int sp;
cin>>sp;
        n1.Insert_spec_SLL(v,sp);
        break;
}
case 4:{
        n1.Del_BEGIN_SLL();
        getch();
        break;
}
case 5:{
        n1.Del_end_SLL();
        break;
}
```

```
case 6:{
        cout<<"\n Enter Data, You want to Delete ";
int sp;
cin>>sp;
        n1.Del_Spec_SLL(sp);
        MessageBox(NULL, message_del, title, MB_ICONINFORMATION |
MB_OK);
        cout<<"\n ==> DELETED SUCCESSFULLY !!! <==\n";
getch();
        break;
}
case 7:{
        n1.Reverse_SLL();
        cout<<"\n ==> Reversed Successfully!!! <==";
        getch();
        break;
}
case 8:{
        n1.Display_SLL_ADRS();
        getch();
        break;
}
case 9:{
        n1.Display_SLL();
        getch();
        break;
}

case 0:{
```

```
                sw=1;
                break;
            }
            default:
                cout<<"\n=> Invalid Choice"<<endl;
            MessageBox(NULL, message, title, MB_ICONINFORMATION | MB_OK);



}//switch SLL

    }while(sw!=1);

    break;
}// SLL  ////////////////////////////////////////////// SLL
//////////////////////////////////////////////////////

case 2:{////////////////////////////////////////////// CLL
//////////////////////////////////////////////////////


    do{
        system("cls");
cout<<"<'''''' Circular Linked-List MENU ''''''>\n";
cout<<" 1. Insert At beginning.\n";
cout<<" 2. Insert At END.\n";
cout<<" 3. Insert At Specific Location.\n";
cout<<" 4. Delete At Beginning.\n";
cout<<" 5. Delete At END.\n";
cout<<" 6. Delete At Specific Location.\n";
```

```
cout<<" 7. Reverse Linked-List.\n";
cout<<" 8. DISPLAY ( ADDRESS ).\n";
cout<<" 9. DISPLAY ( SIMPLE ). \n";
cout<<" 0. To go Back. \n\n";

cout<<"==> Your Choice is ";
int choice;
cin>>choice;
switch(choice){
	case 1:{
		cout<<"\n Enter Value ";
		int v;
		cin>>v;
		n2.Insert_Begin_CLL(v);
		break;
		}
		case 2:{
			cout<<"\n Enter Value ";
		int v;
		cin>>v;
		n2.Insert_END_CLL(v);
			break;
		}
		case 3:{
			cout<<"\n Enter Value you want to add ";
		int v;
		cin>>v;
		cout<<"\n Enter location, you want to put after that value ";
		int sd;
```

```
cin>>sd;
n2.insert_spec_CLL(sd,v);
        break;
}
case 4:{
        n2.Del_Begin_CLL();
                cout<<"\n ==> DELETED SUCCESSFULLY !!! <==\n";
getch();
        break;
}
case 5:{
        n2.Del_END_CLL();
        cout<<"\n ==> DELETED SUCCESSFULLY !!! <==\n";
getch();
        break;
}
case 6:{
        cout<<"\n Enter Data, You want to Delete.\n  ==> You Select ->  ";
int sd;
cin>>sd;
        n2.Del_Spec_CLL(sd);
                cout<<"\n ==> DELETED SUCCESSFULLY !!! <==\n";
getch();
        break;
}
case 7:{
        n2.Reverse_CLL();
        cout<<"\n ==> Reversed Successfully!!! <==";
        getch();
```

```
                break;
        }
        case 8:{
                n2.Display_CLL_ADRS();
                getch();
                break;
        }

        case 9:{
                n2.Display_CLL();
                getch();
                break;
        }
        case 0:{
                cw=1;
                break;
        }
        default:
                cout<<"\n=> Invalid Choice"<<endl;
                        getch();

}

        }while(cw!=1);



        break;
}//CLL  ////////////////////////////////////////////////  CLL
//////////////////////////////////////////////////////
```

```
case 3:{////////////////////////////////////////////// DLL
//////////////////////////////////////////////////////
        do{
                system("cls");
cout<<"<'''''' Doubly Linked-List MENU ''''''>\n";
cout<<" 1. Insert At beginning.\n";
cout<<" 2. Insert At END.\n";
cout<<" 3. Insert At Specific Location.\n";
cout<<" 4. Delete At Beginning.\n";
cout<<" 5. Delete At END.\n";
cout<<" 6. Delete At Specific Location.\n";
cout<<" 7. Reverse Linked-List.\n";
cout<<" 8. DISPLAY ( ADDRESS ).\n";
cout<<" 9. DISPLAY ( SIMPLE ). \n";
cout<<" 0. To go Back. \n\n";

cout<<"==> Your Choice is ";
int choice;
cin>>choice;



switch(choice){
        case 1:{
                cout<<"\n Enter Value ";
                int v;
                cin>>v;
                n3.Insert_Begin_DLL(v);
                break;
                }
```

```
case 2:{
        cout<<"\n Enter Value ";
int v;
cin>>v;
n3.Insert_END_DLL(v);

        break;
}
case 3:{
        cout<<"\n Enter Value ";
        int v;
cin>>v;
cout<<"\n Enter location, you want to put after that value ";
int sp;
cin>>sp;
        n3.Insert_spec_DLL(v,sp);
        break;
}
case 4:{
        n3.Del_BEGIN_DLL();
        getch();
        break;
}
case 5:{
        n3.Del_end_DLL();
        break;
}
case 6:{
        cout<<"\n Enter Data, You want to Delete ";
```

```
int sp;
cin>>sp;
        n3.Del_Spec_DLL(sp);
        MessageBox(NULL, message_del, title, MB_ICONINFORMATION |
MB_OK);
        cout<<"\n ==> DELETED SUCCESSFULLY !!! <==\n";
getch();
        break;
}
case 7:{
        n3.Reverse_DLL();
        cout<<"\n ==> Reversed Successfully!!! <==";
        getch();
        break;
}
case 8:{
        n3.Display_DLL_ADRS();
        getch();
        break;
}
case 9:{
        n3.Display_DLL();
        getch();
        break;
}

case 0:{
        dw=1;
        break;
```

```
                }
                default:
                        cout<<"\n=> Invalid Choice"<<endl;
                MessageBox(NULL, message, title, MB_ICONINFORMATION | MB_OK);

}//switch SLL

        }while(dw!=1);
        break;
}////////////////////////////////////////////////  DLL
//////////////////////////////////////////////////////////
default :{
                cout<<"\n=> Invalid Choice"<<endl;
                MessageBox(NULL, message, title, MB_ICONINFORMATION | MB_OK);
        break;
}
}//switch main
}while(m!=1);

}
```