**Name:- 000**

**Roll No:- 000**

**Practical No:- 1.1**

**Practical Name:- Implementation of programs based on** **Arrays**

#include"iostream.h";

#include"conio.h";

int A[6],size=5,n=0,ele;

void ADD\_END()

{

if(n==size)

{

cout<<endl<<"List is full";

return;

}

n=n+1;

A[n]=ele;

cout<<endl<<"added succesfully !";

}

int DEL\_END()

{

if(n>0)

{

ele=A[n];

n=n-1;

return(ele);

}

else

{

cout<<endl<<"List is empty";

return NULL;

}

}

void LIST\_ALL()

{

cout<<endl;

if(n>0)

for(int i=1;i<=n;i++)

cout<<A[i]<<" ";

else

cout<<endl<<"List is empty";

}

void MENU()

{

do

{

int option;

cout<<endl<<"-----------------------";

cout<<endl<<"1.ADD";

cout<<endl<<"2.DELETE";

cout<<endl<<"3.LIST all the element";

cout<<endl<<"4.EXIT Menu";

cout<<endl<<"-----------------------";

cout<<endl<<"Enter Your Choice:";

cin>>option;

cout<<option;

switch(option)

{

case 1:

cout<<endl<<"Enter Element to Add";

cin>>ele;

ADD\_END();

break;

case 2:

cout<<endl<<"Selected Delete";

DEL\_END();

break;

case 3:

cout<<endl<<"Selected List All";

LIST\_ALL();

break;

case 4:

cout<<endl<<"Exit Menu";

return;

default:

cout<<endl<<"Selected Invalid";

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch();

}

**Name:- 000**

**Roll No:- 000**

**Practical No:- 1.3.3**

**Practical Name:-** Implementation of programs based on **Infix to Postfix**.

#include"iostream.h"

#include"conio.h"

#include"string.h"

class CONVERT

{

char infix[99],postfix[99],s[99];

int i,p,top;

public:

CONVERT();

int PRECEDANCE(char);

void POST();

void DISPLAY();

};

CONVERT::CONVERT()

{

top=1;

i=p=0;

cout<<"\n Enter infix Expression";

cin>>infix;

strcat(infix,")");

s[++top]='(';

}

int CONVERT::PRECEDANCE(char ch)

{

switch(ch)

{

case'^' : return 3;

case'\*' : return 2;

case'/' : return 2;

case'+' : return 1;

case'-' : return 1;

default : return 0;

}

}

void CONVERT::POST()

{

char ch;

while(top!=-1)

{

ch=infix[i++];

if((ch>'A'&&ch<'Z')||(ch>='a'&&ch<='z')||(ch>='1'&&ch<='9'))

postfix[p++]=ch;

else if(ch=='(')

s[++top]=ch;

else if(ch=='+'||ch=='-'||ch=='\*'||ch=='/'||ch=='^')

{

while(PRECEDANCE(ch)<=PRECEDANCE(s[top]))

postfix[p++]=s[top--];

s[++top]=ch;

}

else if(ch==')')

{

while(s[top]!='(')

postfix[p++]=s[top--];

top--;

}

else

cout<<"\n wrong string";

}

postfix[p]='\0';

}

void CONVERT::DISPLAY()

{

cout<<"\n Postfix Expression is: ";

}

void main()

{

clrscr();

CONVERT c;

c.POST();

c.DISPLAY();

getch();

}

**Name:- 000**

**Roll No:- 000**

**Practicle No:- 1.3.4**

**Practicle Name:** Implementation of programs to **Evaluate Postfix** Expression.

#include"iostream.h"

#include"conio.h"

#include"ctype.h"

#include"string.h"

class EXPRESSION

{

char POST[100];

public:

EXPRESSION();

void READ\_POSTFIX();

void EVALUATE();

};

EXPRESSION::EXPRESSION()

{

POST[0]= '\0';

}

void EXPRESSION::READ\_POSTFIX()

{

cout<<"\n Enter post fix expression: ";

cin>>POST;

}

void EXPRESSION::EVALUATE()

{

int STK[50], top=0;

int temp, result;

int i = 0;

while(POST[i] != '\0')

{

if(isalpha(POST[i]))

{

cout<<endl<<"Enter values of "<<POST[i]<<" ";

cin>>temp;

top=top+1;

STK[top] = temp;

}

else

{

int right = STK[top]; top = top - 1;

int left = STK[top]; top = top - 1;

if(POST[i]=='+')

result = left + right;

else

if(POST[i]=='-')

result = left - right;

else

if(POST[i]=='\*')

result = left \* right;

else

if(POST[i]=='/')

result = left / right;

top=top+1;

STK[top] = result;

}

i++;

}

cout<<endl<<"Result of expression: "<<result;

}

void main()

{

clrscr();

EXPRESSION obj;

obj.READ\_POSTFIX();

obj.EVALUATE();

getch();

}

Name:- 000

Roll No:- 000

Practical Name:- Implementation of programs based on **Queue** (using Array)

#include<iostream.h>

#include<conio.h>

class QUEUE

{

int A[20],size,rear,front;

public:

QUEUE(int);

void ADD\_QUE(int);

int DEL\_QUE();

void LIST\_QUE();

};

QUEUE::QUEUE(int par)

{

size=par;

rear = front = 0;

}

void QUEUE::ADD\_QUE(int ele)

{

if(rear==size)

cout<<endl<<"Queue is full";

else

{

if(front==0)

front=1;

rear = rear + 1;

A[rear] = ele;

}

}

int QUEUE::DEL\_QUE()

{

if(front==0)

{

cout<<endl<<"Queue is empty";

return NULL;

}

else

{

int ele = A[front];

if(front == rear)

rear = front = 0;

else

front = front + 1;

return ele;

}

}

void QUEUE::LIST\_QUE()

{

if(front==0)

cout<<endl<<"Queue is empty";

else

{

cout<<endl<<"Queue elements are :";

for(int i=front;i<=rear;i++)

cout<<A[i]<<" ";

}

}

void MENU()

{

int n,ch,ele;

cout<<endl<<"Enter the size of Queue :";

cin>>n;

QUEUE obj(n);

do{

cout<<endl<<"\_\_\_\_\_\_\_\_\_\_MENU\_\_\_\_\_\_\_\_\_\_";

cout<<endl<<"1)ADD\_QUE";

cout<<endl<<"2)DEL\_QUE";

cout<<endl<<"3)VIEW\_ALL";

cout<<endl<<"4)EXIT";

cout<<endl<<"Enter Your Choice: ";

cin>>ch;

switch(ch)

{

case 1:

cout<<endl<<"Selected ADD\_QUE";

cout<<endl<<"Enter element to Add:";

cin>>ele;

obj.ADD\_QUE(ele);

break;

case 2:

cout<<endl<<"Selected DEL\_QUE";

obj.DEL\_QUE();

break;

case 3:

cout<<endl<<"Selected VIEW\_ALL";

obj.LIST\_QUE();

break;

case 4:

cout<<"EXIT";

return;

default:

cout<<endl<<"Invalid Choice";

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch();

}

**Name:- 000**

**Roll No:-000**

**Assignment Name:-Implementation of program based on Queue using Linked List.**

**-----------------------------------------------------------------------------------------------------**

#include "iostream.h"

#include "conio.h"

class NODE

{

public:

int data;

NODE \*link;

};

class LIST

{

NODE \*front;

NODE \*rear;

public:

LIST();

void ADD\_REAR(int);

int DEL\_FRONT();

void VIEW();

int IS\_EMPTY();

};

LIST::LIST()

{

front = NULL;

rear = NULL;

}

void LIST::ADD\_REAR(int ele)

{

NODE \*NN;

NN= new NODE();

NN->data = ele;

NN->link = NULL;

if(front == NULL)

front=rear=NN;

else

{

NODE \*ptr;

ptr= rear;

while(ptr->link != NULL)

{

ptr= ptr->link;

}

ptr->link = NN;

}

}

int LIST::DEL\_FRONT()

{

if(front == NULL)

{

cout<<endl<<"List is empty";

return NULL;

}

int ele = front->data;

NODE \* TEMP = front;

front = front->link;

delete TEMP;

return ele;

}

int LIST::IS\_EMPTY()

{

if(front == NULL)

return 1;

else

return 0;

}

void LIST::VIEW()

{

if(front == NULL)

{

cout<<endl<<"List is empty";

return;

}

NODE \*ptr = front;

cout<<endl<<"List elements are : ";

while(ptr != NULL)

{

cout<<ptr-> data<<" ";

ptr=ptr->link;

}

}

void MENU()

{

int ele, opt, pos;

LIST obj;

do

{

cout<<endl<<"1 Add at rear";

cout<<endl<<"2 Delete from Front";

cout<<endl<<"3 View All";

cout<<endl<<"4 Exit Menu";

cout<<endl<<"================\n";

cout<<endl<<"Enter your choice : ";

cin>>opt;

switch(opt)

{

case 1:

cout<<endl<<"Enter element : ";

cin>>ele;

obj.ADD\_REAR(ele);

obj.VIEW();

break;

case 2:

if(!obj.IS\_EMPTY())

{

ele = obj.DEL\_FRONT();

cout<<endl<<"Delted element = "<<ele;

obj.VIEW();

}

break;

case 3:

obj.VIEW();

break;

case 4:

return;

default:

cout<<endl<<"invalid input";

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch();

}

**Name:- 000**

**Roll No:- 000**

**Practical Name:-** **Implementation of programs based on Circular-Queue (using Array)**

#include "iostream.h"

#include "conio.h"

class QUEUE

{

int A[20],size,rear,front;

public:

QUEUE(int);

void ADD\_QUE(int);

int DEL\_QUE();

void LIST\_QUE();

};

QUEUE::QUEUE(int par)

{

size= par;

rear = front = 0;

}

void QUEUE::ADD\_QUE(int ele)

{

if( (front==1 && rear==size) || (rear+1==front) )

cout<<endl<<"Que is full";

else

{

if(front==0)

front=1;

if(rear==size)

rear=0;

rear = rear + 1;

A[rear] = ele;

}

}

int QUEUE::DEL\_QUE()

{

if(front==0)

{

cout<<endl<<"Que is empty";

return NULL;

}

else

{

int ele = A[front];

if(front == rear)

{

rear =front = 0;

}

else

{

if(front==size)

{

front=0;

}

front = front + 1;

}

return ele;

}

}

void QUEUE::LIST\_QUE()

{

if(front==0)

cout<<endl<<"Que is empty";

else

{

cout<<endl<<"Que elements are : ";

if(front<=rear)

{

for(int i= front;i<=rear;i++)

cout<<A[i]<<" ";

}

else

{

for(int i= front;i<=size;i++)

cout<<A[i]<<" ";

for(i=1;i<=rear;i++)

cout<<A[i]<<" ";

}

}

}

void MENU()

{

int ele,n,opt;

cout<<endl<<"Enter the size of Queue : ";

cin>>n;

QUEUE obj(n);

do

{

cout<<endl<<"\n1 Add element";

cout<<endl<<"2 Delete element";

cout<<endl<<"3 View Queue";

cout<<endl<<"4 Exit Menu";

cout<<endl<<"================\n";

cout<<endl<<"Enter your choice : ";

cin>>opt;

switch(opt)

{

case 1:

cout<<endl<<"Enter ele to add : ";

cin>>ele;

obj.ADD\_QUE(ele);

break;

case 2:

ele = obj.DEL\_QUE();

if(ele)

cout<<endl<<"Deleted ele = "<<ele;

break;

case 3:

obj.LIST\_QUE();

break;

case 4:

return;

default:

cout<<endl<<"invalid input";

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch();

}

**Name: - 000**

**Roll No: - 000**

**Practical: - Implement of program base on priority Queue.**

#include"iostream.h"

#include"conio.h"

class PQUEUE

{

int \*A, size, front, rear;

public:

PQUEUE(int);

void ADD\_PQUEUE(int ele);

int DEL\_PQUEUE();

void VIEW\_ALL\_PQUEUE();

};

PQUEUE::PQUEUE(int par)

{

size=par;

front=0;

rear=0;

}

void PQUEUE::ADD\_PQUEUE(int ele)

{

if(front==0)

{

front=1;

rear=1;

A[rear]=ele;

return;

}

for(int i=front;i<=rear;i++)

{

if(A[i]>ele)

break;

}

if(i<=rear)

{

for(int j=rear;j>=i;j--)

{

A[j+1]=A[j];

}

}

A[i]=ele;

rear=rear+1;

}

int PQUEUE::DEL\_PQUEUE()

{

if(front==0)

{

cout<<endl<<"Queue is empty";

return NULL;

}

else

{

int ele=A[front];

if(front==rear)

front=rear=0;

else

{

if(front==size)

front=0;

else

front=front+1;

}

return(ele);

}

}

void PQUEUE::VIEW\_ALL\_PQUEUE()

{

if(front==0)

cout<<endl<<"Queue is empty";

else

{

if(front<=rear)

{

for(int i=front; i<=rear; i++)

cout<<endl<<A[i]<<"";

}

else

{

for(int i=front;i<=size;i++)

cout<<endl<<A[i]<<"";

for(i=1;i<=rear;i++)

cout<<endl<<A[i]<<" ";

}

}

}

void MENU()

{

int option, ele, n;

cout<<endl<<"Enter the size of the Queue ";

cin>>n;

PQUEUE obj(n);

do

{

cout<<endl<<"1.ADD\_PQUEUE";

cout<<endl<<"2.DEL\_PQUEUE";

cout<<endl<<"3.VIEW\_ALL\_PQUEUE";

cout<<endl<<"4.EXIT";

cout<<endl<<"Enter the option ";

cin>>option;

switch(option)

{

case 1:

cout<<endl<<"Enter to the add element ";

cin>>ele;

obj.ADD\_PQUEUE(ele);

break;

case 2:

ele=obj.DEL\_PQUEUE();

if(ele)

cout<<endl<<"Delete the element "<<ele;

break;

case 3:

obj.VIEW\_ALL\_PQUEUE();

break;

case 4:

cout<<endl<<"Exit";

return;

default:

cout<<endl<<"Invalid option";

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch();

}

**Name: - 000**

**Roll No: - 000**

**Assignment: - Implementation of program for base on DeQueue.**

#include "iostream.h"

#include "conio.h"

class QUEUE

{

int \*A,size,rear,front;

public:

QUEUE(int);

void ADD\_REAR(int);

int DEL\_FRONT();

int DEL\_REAR();

void LIST\_QUE();

};

QUEUE::QUEUE(int par)

{

size= par;

A =new int[size+1];

rear = front = 0;

}

void QUEUE::ADD\_REAR(int ele)

{

if(rear==size)

cout<<endl<<"Que is full";

else

{

if(front==0)

front=1;

rear = rear + 1;

A[rear] = ele;

}

}

int QUEUE::DEL\_FRONT()

{

if(front==0)

{

cout<<endl<<"Que is empty";

return NULL;

}

else

{

int ele = A[front];

if(front == rear)

rear =front = 0;

else

front = front + 1;

return ele;

}

}

int QUEUE::DEL\_REAR()

{

if(front==0)

{

cout<<endl<<"Que is empty";

return NULL;

}

else

{

int ele = A[rear];

if(rear == front)

front = rear = 0;

else

rear = rear -1;

return ele;

}

}

void QUEUE::LIST\_QUE()

{

if(front==0)

cout<<endl<<"Que is empty";

else

{

cout<<endl<<"Que elements are : ";

for(int i= front;i<=rear;i++)

cout<<A[i]<<" ";

}

}

void MENU()

{

int n,opt,ele;

cout<<endl<<"Enter the size of Queue : ";

cin>>n;

QUEUE obj(n);

do

{

cout<<endl<<"1 ADD @ REAR";

cout<<endl<<"2 DEL from FRONT";

cout<<endl<<"3 DEL from REAR";

cout<<endl<<"4 EXIT";

cout<<endl<<"================\n";

cout<<endl<<"Enter your choice : ";

cin>>opt;

switch(opt)

{

case 1:

cout<<endl<<"Enter ele";

cin>>ele;

obj.ADD\_REAR(ele);

obj.LIST\_QUE();

break;

case 2:

ele = obj.DEL\_FRONT();

if(ele)

cout<<endl<<"Delted ele = "<<ele;

obj.LIST\_QUE();

break;

case 3:

ele = obj.DEL\_REAR();

if(ele)

cout<<endl<<"Delted ele = "<<ele;

obj.LIST\_QUE();

break;

case 4:

return;

default:

cout<<endl<<"invalid input";

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch();

}

**Name:- 000**

**Roll No:- 000**

**Practical: - Implementation of Program for Single Liked List.**

#include "iostream.h"

#include "conio.h"

class NODE

{

public:

int data;

NODE \*link;

};

class LIST

{

NODE \*start;

public:

LIST();

void ADD\_FIRST(int); // 1

void ADD\_LAST(int); // 2

void ADD\_POS(int,int); // 3

int DEL\_FIRST(); // 4

int DEL\_LAST(); // 5

int DEL\_POS(int); // 6

void VIEW(); // 7

int IS\_EMPTY();

};

LIST::LIST()

{

start = NULL;

}

void LIST::ADD\_FIRST(int ele)

{

//------- (A) create new node --------------

NODE \*NN;

NN= new NODE(); // allocate new node

if(NN == NULL)

{

cout<<endl<<"List is full";

return;

}

//----- (B) fill up new node -------------

NN->data = ele;

NN->link = NULL;

//---- (C) set the links -----------------

if(start==NULL) //case - II Not Full - Empty

start=NN;

else

{

NN->link = start;//sets null in case of list initially empty

start = NN;

}

}

void LIST::ADD\_LAST(int ele)

{

//----- (A) create new node --------------

NODE \*NN;

NN= new NODE();

{

cout<<endl<<"List is full";

return;

} //----- (B) fill up new node -------------

NN->data = ele;

NN->link = NULL;

//---- (C) set the links -----------------

if(start == NULL)

start=NN;

else

{

NODE \*ptr;

ptr= start;

while(ptr->link != NULL)

{

ptr= ptr->link;

}

ptr->link = NN;

}

}

void LIST::ADD\_POS(int ele,int pos)

{

//----- (A) create new node --------------

NODE \*NN;

NN= new NODE(); // allocate new node

//----- (B) fill up new node -------------

NN->data = ele;

NN->link = NULL;

//---- (C) set the links -----------------

if(pos == 1)

{

NN->link = start;

start = NN;

}

else

{

int count = 1;

// ---- using pair of pointer variables ----

NODE \*ptr = start;

NODE \*prev = NULL;

while(count < pos)

{

prev = ptr;

ptr = ptr->link;

count = count + 1;

}

NN->link = ptr;

prev->link = NN;

}

}

int LIST::DEL\_FIRST()

{

// def of function

if(start == NULL)

{

cout<<endl<<"List is empty";

return NULL;

}

int ele = start->data;

NODE \* TEMP = start;

start = start->link;

delete TEMP;

return ele;

}

int LIST::DEL\_LAST()

{

// def of function

if(start == NULL)

{

cout<<endl<<"List is empty";

return NULL;

}

// use pair of pointers

NODE \*ptr = start;

NODE \*prev = NULL;

while(ptr->link != NULL) //move the ptrs up to last node

{

prev = ptr;

ptr = ptr ->link;

}

int ele = ptr->data;

NODE \*TEMP = ptr;

if (prev ==NULL) // only one ele in list

start =NULL;

else

prev->link = NULL; // set the link

delete TEMP;

return ele;

}

int LIST::DEL\_POS(int pos)

{

// def of function

NODE \*TEMP;

int ele;

if(start == NULL)

{

cout<<endl<<"List is empty";

return NULL;

}

else

{

if(pos==1)

{

ele = start->data;

TEMP = start;

start = start->link;

}

else

{

NODE \*ptr, \*prev;

int count = 1;

ptr = start; prev = NULL;

while(count<pos)

{

prev = ptr;

ptr = ptr->link;

count = count+1;

}

ele = ptr->data;

TEMP = ptr;

prev->link = ptr->link;

}

delete TEMP;

return ele;

}

}

int LIST::IS\_EMPTY()

{

if(start == NULL)

return 1;

else

return 0;

}

void LIST::VIEW()

{

if(start == NULL)

{

cout<<endl<<"List is empty";

return;

}

NODE \*ptr = start;

cout<<endl<<"List elements are : ";

while(ptr != NULL)

{

cout<<ptr-> data<<" ";

ptr=ptr->link;

}

}

void MENU()

{

int ele, opt, pos;

LIST obj;

do

{

cout<<endl<<"1 Add at First";

cout<<endl<<"2 Add at Last";

cout<<endl<<"3 Add at Position";

cout<<endl<<"4 Delete from First";

cout<<endl<<"5 Delete from Last";

cout<<endl<<"6 Delete from Position";

cout<<endl<<"7 List All";

cout<<endl<<"8 Exit Menu";

cout<<endl<<"================\n";

cout<<endl<<"Enter your choice : ";

cin>>opt;

switch(opt)

{

case 1:

cout<<endl<<"Enter element : ";

cin>>ele;

obj.ADD\_FIRST(ele);

obj.VIEW();

break;

case 2:

cout<<endl<<"Enter element : ";

cin>>ele;

obj.ADD\_LAST(ele);

obj.VIEW();

break;

case 3:

cout<<endl<<"Enter element : ";

cin>>ele;

cout<<endl<<"Enter position : ";

cin>>pos;

obj.ADD\_POS(ele,pos);

obj.VIEW();

break;

case 4:

if(!obj.IS\_EMPTY())

{

ele = obj.DEL\_FIRST();

cout<<endl<<"Delted element = "<<ele;

obj.VIEW();

}

break;

case 5:

if(!obj.IS\_EMPTY())

{

ele = obj.DEL\_LAST();

cout<<endl<<"Delted element = "<<ele;

obj.VIEW();

}

break;

case 6:

if(!obj.IS\_EMPTY())

{

cout<<endl<<"Enter position : ";

cin>>pos;

ele = obj.DEL\_POS(pos);

cout<<endl<<"Delted element = "<<ele;

obj.VIEW();

}

break;

case 7:

obj.VIEW();

break;

case 8:

return;

default:

cout<<endl<<"invalid input";

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch();

}

Name – 000

Roll no – 000

Practical name-CLL

Practical no-

// Implementation of Circular Linked List

#include "iostream.h"

#include "conio.h"

class NODE

{

public:

int data;

NODE \*next;

};

class LIST

{

// data

NODE \*start,\*end;

public:

// operations

LIST();

void ADD\_FIRST(int); // 1

void ADD\_LAST(int); // 2

int DEL\_FIRST(); // 3

int DEL\_LAST(); // 4

void VIEW\_FWD(); // 5

int IS\_EMPTY();

};

LIST::LIST()

{

// def of function

start = NULL;

end = NULL;

}

void LIST::ADD\_FIRST(int ele)

{

// def of function

//case - I Check List is full?

//----- (A) create new node --------------

NODE \*NN;

NN= new NODE(); // allocate new node

/\*if(NN == NULL)

{

cout<<endl<<"List is full";

return;

}\*/

//----- (B) fill up new node -------------

NN->data = ele;

NN->next = NULL;

//---- (C) set the links -----------------

if(start==NULL) //case - II Not Full - Empty

{

start = NN;

end = NN;

end->next = NN; //IMP

}

else //case - III Not Full - Not Empty

{

NN->next = start;//sets null in case of list initially empty

start = NN;

end->next = NN; //start

}

}

void LIST::ADD\_LAST(int ele)

{

// def of function

//case - I List is full

//----- (A) create new node --------------

NODE \*NN;

NN= new NODE(); // allocate new node

/\*if(NN == NULL) // optional

{

cout<<endl<<"List is full";

return;

}\*/

//----- (B) fill up new node -------------

NN->data = ele;

NN->next = NULL;

//---- (C) set the links -----------------

if(start==NULL) //case - II Not Full - Empty

{

NN->next = NN;//IMP

start = NN;

end = NN;

}

else

{

end->next = NN;

NN->next = start;

end = NN;

}

}

int LIST::DEL\_FIRST()

{

// def of function

if(start == NULL)

{

cout<<endl<<"List is empty";

return NULL;

}

else

{

int ele = start->data;

NODE \* TEMP = start;

if(start->next == start) // only one node in list

{

start = NULL;

end = NULL;

}

else

{

start = start->next;

end->next = start;

}

delete TEMP;

return ele;

}

}

int LIST::DEL\_LAST()

{

// def of function

if(start == NULL)

{

cout<<endl<<"List is empty";

return NULL;

}

else

{

int ele = end->data;

NODE \* TEMP = end;

if(start->next == start) // only one node in list

{

start = NULL;

end = NULL;

}

else

{

NODE \*prev = start;

while(prev->next != end)

{

prev = prev->next;

}

prev->next=start;

end=prev;

}

delete TEMP;

return ele;

}

}

int LIST::IS\_EMPTY()

{

// def of function

if(start == NULL)

return 1;

else

return 0;

}

void LIST::VIEW\_FWD()

{

// def of function

if(start == NULL)

{

cout<<endl<<"List is empty";

return;

}

cout<<endl<<"List elements are : ";

cout<<start->data<<" ";

NODE \*ptr = start->next;

while(ptr != start )

{

cout<<ptr-> data<<" ";

ptr=ptr->next;

}

}

void MENU()

{

int ele, opt, pos;

LIST obj;

do

{

cout<<endl<<"================\n";

cout<<endl<<"1 Add at First";

cout<<endl<<"2 Add at Last";

cout<<endl<<"3 Delete from First";

cout<<endl<<"4 Delete from Last";

cout<<endl<<"5 List All (FWD)" ;

cout<<endl<<"6 Exit";

cout<<endl<<"================\n";

cout<<endl<<"Enter your choice : ";

cin>>opt;

switch(opt)

{

case 1:

cout<<endl<<"Enter element : ";

cin>>ele;

obj.ADD\_FIRST(ele);

obj.VIEW\_FWD();

break;

case 2:

cout<<endl<<"Enter element : ";

cin>>ele;

obj.ADD\_LAST(ele);

obj.VIEW\_FWD();

break;

case 3:

if(!obj.IS\_EMPTY())

{

ele = obj.DEL\_FIRST();

cout<<endl<<"Delted element = "<<ele;

obj.VIEW\_FWD();

}

else

cout<<endl<<"List is empty";

break;

case 4:

if(!obj.IS\_EMPTY())

{

ele = obj.DEL\_LAST();

cout<<endl<<"Delted element = "<<ele;

obj.VIEW\_FWD();

}

else

cout<<endl<<"List is empty";

break;

case 5:

obj.VIEW\_FWD();

break;

case 6:

return;

default:

cout<<endl<<"invalid input";

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch();

}

**Name:- 000**

**Roll No:-000**

**Assignment No:-**

**Assignment Title:-Implementation of program based on the Doubly linked list.**

**---------------------------------------------------------------------------------------------------------------------**

#include "iostream.h"

#include "conio.h"

class NODE

{

public:

int data;

NODE \*next;

NODE \*prev;

};

class LIST

{

NODE \*start;

public:

LIST();

void ADD\_FIRST(int);

void ADD\_LAST(int);

int DEL\_FIRST();

int DEL\_LAST();

void VIEW\_BACK();

void VIEW\_FORW();

int IS\_EMPTY();

};

LIST::LIST()

{

start = NULL;

}

void LIST::ADD\_FIRST(int ele)

{

NODE \*NN;

NN= new NODE();

if(NN == NULL)

{

cout<<endl<<"List is full";

return;

}

NN->data = ele;

NN->next = NULL;

NN->prev = NULL;

NN->next = start;

if(start!=NULL)

{

start->prev=NN;

}

start = NN;

}

void LIST::ADD\_LAST(int ele)

{

NODE \*NN;

NN= new NODE();

NN->data = ele;

NN->next = NULL;

if(start == NULL)

start=NN;

else

{

NODE \*ptr;

ptr= start;

while(ptr->next != NULL)

{

ptr= ptr->next;

}

NN->prev =ptr;

ptr->next = NN;

}

}

int LIST::DEL\_FIRST()

{

if(start == NULL)

{

cout<<endl<<"List is empty";

return NULL;

}

int ele = start->data;

NODE \* TEMP = start;

//cout<<endl<<"Deleted Elements are";

start = start->next;

start-> prev=NULL;

delete TEMP;

return ele;

}

int LIST::DEL\_LAST()

{

if(start == NULL)

{

cout<<endl<<"List is empty";

return NULL;

}

NODE \*ptr1 = start;

NODE \*ptr2 = NULL;

while(ptr1->next != NULL)

{

ptr2 = ptr1;

ptr1 = ptr1 ->next;

}

int ele = ptr1->data;

NODE \*TEMP = ptr1;

if (ptr2!=NULL)

ptr2->next = NULL;

else

start = NULL;

delete TEMP;

return ele;

}

int LIST::IS\_EMPTY()

{

if(start == NULL)

return 1;

else

return 0;

}

void LIST::VIEW\_FORW()

{

if(start == NULL)

{

cout<<endl<<"List is empty";

return;

}

NODE \*ptr = start;

cout<<endl<<"List elements are : ";

while(ptr != NULL)

{

cout<<ptr-> data<<" ";

ptr=ptr->next;

}

}

void LIST::VIEW\_BACK()

{

if(start == NULL)

{

cout<<endl<<"List is empty ";

return ;

}

NODE \*ptr = start;

cout<<endl<<"List elements are :";

while(ptr-> next!= NULL)

{

ptr=ptr->next;

}

while(ptr != NULL)

{

cout<<ptr->data<<" ";

ptr=ptr->prev;

}

}

void MENU()

{

int ele, opt, pos;

LIST obj;

do

{

cout<<endl<<"1 Add at First";

cout<<endl<<"2 Add at Last";

cout<<endl<<"3 Delete from First";

cout<<endl<<"4 Delete from Last";

cout<<endl<<"5 View Backword";

cout<<endl<<"6 View Forward";

cout<<endl<<"7 Exit Menu";

cout<<endl<<"================\n";

cout<<endl<<"Enter your choice : ";

cin>>opt;

switch(opt)

{

case 1:

cout<<endl<<"Enter element : ";

cin>>ele;

obj.ADD\_FIRST(ele);

obj.VIEW\_FORW();

obj.VIEW\_BACK();

break;

case 2:

cout<<endl<<"Enter element : ";

cin>>ele;

obj.ADD\_LAST(ele);

obj.VIEW\_FORW();

obj.VIEW\_BACK();

break;

case 3:

if(!obj.IS\_EMPTY())

{

ele = obj.DEL\_FIRST();

cout<<endl<<"Delted element = "<<ele;

obj.VIEW\_FORW();

}

else

{

cout<<endl<<"List is empty";

}

break;

case 4:

if(!obj.IS\_EMPTY())

{

ele = obj.DEL\_LAST();

cout<<endl<<"Delted element = "<<ele;

obj.VIEW\_FORW();

}

else

{

cout<<endl<<"List is empty ";

}

break;

case 5:

obj.VIEW\_BACK();

break;

case 6:

obj.VIEW\_FORW();

break;

case 7:

return;

default:

cout<<endl<<"invalid input";

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch();

}

**Name: - 000**

**Roll No:-000**

**Assignment: - Implementation of Polynomial Addition / Subtraction (using Array)**

#include "iostream.h"

#include "conio.h"

class POLYEXPR

{

int PE1[10],PE2[10],PE3[10];

int order;

public:

POLYEXPR(int);

void READ\_POLYEXPR1();

void READ\_POLYEXPR2();

void ADD\_POLYEXPR();

void SUB\_POLYEXPR();

void VIEW\_POLYEXPR();

};

POLYEXPR::POLYEXPR(int para)

{

order = para;

}

void POLYEXPR::READ\_POLYEXPR1()

{

cout<<endl<<"Enter poly Exp 1 : ";

for (int i=order;i>=0;i--)

{

cout<<endl<<"Enter Coeff of X^"<<i<<" : ";

cin>>PE1[i];

}

}

void POLYEXPR::READ\_POLYEXPR2()

{

cout<<endl<<"Enter poly Exp 2 : ";

for (int i=order;i>=0;i--)

{

cout<<endl<<"Enter Coeff of X^"<<i<<" : ";

cin>>PE2[i];

}

}

void POLYEXPR::ADD\_POLYEXPR()

{

for (int i=order;i>=0;i--)

PE3[i]=PE1[i]+PE2[i];

}

void POLYEXPR::SUB\_POLYEXPR()

{

for (int i=order;i>=0;i--)

PE3[i]=PE1[i]-PE2[i];

}

void POLYEXPR::VIEW\_POLYEXPR()

{

cout<<endl<<"Poly Exp 1 : ";

for (int i=order;i>=0;i--)

{

if(i>=2 && PE1[i] !=0)

{

if(PE1[i]==1)

cout<<"X^"<<i<<" + ";

else

cout<<PE1[i]<<"X^"<<i<<" + ";

}

else

{

if(i==1 &&PE1[i] !=0)

{

if(PE1[i]==1)

cout<<"X + ";

else

cout<<PE1[i]<<"X + ";

}

else

{

if(PE1[i] !=0)

cout<<PE1[i];

}

}

}

}

void main()

{

int ord;

clrscr();

cout<<endl<<"Enter max order of Poly Expression : ";

cin>>ord;

POLYEXPR obj(ord);

obj.READ\_POLYEXPR1();

obj.VIEW\_POLYEXPR();

getch();

}

**Name: - 000**

**Roll No: - 000**

**Assignment: - Implementation of Polynomial Addition / Subtraction (using Linked List)**

#include "iostream.h"

#include "conio.h"

class NODE

{

public:

int coeff;

int power;

NODE \*link;

};

class POLYEXPR

{

NODE \*PE1,\*PE2,\*PE3;

int order;

public:

POLYEXPR(int);

void READ\_POLYEXPR1();

void READ\_POLYEXPR2();

void ADD\_POLYEXPR();

void SUB\_POLYEXPR();

void VIEW\_POLYEXPR();

};

POLYEXPR::POLYEXPR(int para)

{

order = para;

PE1 = NULL;

PE2 = NULL;

PE3 = NULL;

}

void POLYEXPR::READ\_POLYEXPR1()

{

NODE \*last = NULL;

cout<<endl<<"Enter poly Exp 1 : ";

for (int i=order;i>=0;i--)

{

// ---create new node ---------

NODE \*NN = new NODE();

// --- fill up new node -------

cout<<endl<<"Enter Coeff of X^"<<i<<" : ";

cin>>NN->coeff;

NN->power = i;

// --- set the links (ADD at LAST) ---

if(PE1 == NULL)

PE1 = last = NN;

else

{

last->link = NN;

last = NN;

}

}

}

void POLYEXPR::READ\_POLYEXPR2()

{

NODE \*last = NULL;

cout<<endl<<"Enter poly Exp 1 : ";

for (int i=order;i>=0;i--)

{

// ---create new node ---------

NODE \*NN = new NODE();

// --- fill up new node -------

cout<<endl<<"Enter Coeff of X^"<<i<<" : ";

cin>>NN->coeff;

NN->power = i;

// --- set the links (ADD at LAST) ---

if(PE2 == NULL)

PE2 = last = NN;

else

{

last->link = NN;

last = NN;

}

}

}

void POLYEXPR::ADD\_POLYEXPR()

{

NODE \*last = NULL;

NODE \*ptr1 =PE1;

NODE \*ptr2 =PE2;

for (int i=order;i>=0;i--)

{

// ---create new node ---------

NODE \*NN = new NODE();

// --- fill up new node -------

cout<<endl<<"Enter Coeff of X^"<<i<<" : ";

NN->coeff = ptr1->coeff + ptr2->coeff ;

NN->power = i;

// --- set the links (ADD at LAST) ---

if(PE3 == NULL)

PE3 = last = NN;

else

{

last->link = NN;

last = NN;

}

ptr1 = ptr1->link;

ptr2 = ptr2->link;

}

}

void POLYEXPR::SUB\_POLYEXPR()

{

NODE \*last = NULL;

NODE \*ptr1 =PE1;

NODE \*ptr2 =PE2;

for (int i=order;i>=0;i--)

{

// ---create new node ---------

NODE \*NN = new NODE();

// --- fill up new node -------

cout<<endl<<"Enter Coeff of X^"<<i<<" : ";

NN->coeff = ptr1->coeff - ptr2->coeff ;

NN->power = i;

// --- set the links (ADD at LAST) ---

if(PE3 == NULL)

PE3 = last = NN;

else

{

last->link = NN;

last = NN;

}

ptr1 = ptr1->link;

ptr2 = ptr2->link;

}

}

void POLYEXPR::VIEW\_POLYEXPR()

{

NODE \*ptr = PE3;

cout<<endl<<"Poly Exp 1 : ";

for (int i=order;i>=0;i--)

{

cout<<ptr->coeff<<"X^"<<i<<" + ";

ptr= ptr->link;

}

}

void main()

{

int ord;

clrscr();

cout<<endl<<"Enter max order of Poly Expression : ";

cin>>ord;

POLYEXPR obj(ord);

obj.READ\_POLYEXPR1();

obj.READ\_POLYEXPR2();

obj.ADD\_POLYEXPR();

//obj.SUB\_POLYEXPR();

obj.VIEW\_POLYEXPR();

getch();

}

Name:- 000

Roll No:- 000

Title:-Implementation of program base on binary search.

--------------------------------------------------------------------------------------------------------------------------------------

#include "iostream.h"

#include "conio.h"

#include "stdlib.h"

class LIST

{

int \*A,size;

public:

LIST(int);

void SET\_LIST();

void VIEW\_LIST();

void BUBBLE\_SORT();

int BINARY\_SEARCH(int);

};

LIST::LIST(int par)

{

size=par;

A =new int[size+1];

}

void LIST::SET\_LIST()

{

//cout<<endl<<"Enter list elements : ";

for(int i=1;i<=size;i++)

A[i]=random(1000);

}

void LIST::VIEW\_LIST()

{

cout<<"List elements are : ";

for(int i=1;i<=size;i++)

cout<<A[i]<<" ";

}

void LIST::BUBBLE\_SORT()

{

for(int i=1;i<=size-1;i++)

for(int j=1;j<=size-i;j++)

if(A[j]>A[j+1])

{

int temp = A[j];

A[j] = A[j+1];

A[j+1] = temp;

}

}

int LIST::BINARY\_SEARCH(int x)

{

int low=1,high=size,mid;

while(low<=high)

{

mid = (low+high)/2;

if(A[mid]==x)

return mid;

else

if(x < A[mid])

high = mid-1;

else

low = mid+1;

}

return 0;

}

void main()

{

int n,ele,pos;

clrscr();

cout<<"\n Enter size of array : ";

cin>>n;

LIST obj(n);

obj.SET\_LIST();

cout<<endl<<"List before sorting : \n";

obj.VIEW\_LIST();

cout<<endl<<"Enter element to search:";

cin>>ele;

pos = obj.BINARY\_SEARCH(ele);

if(pos !=0)

cout<<endl<<ele<<"fount at"<<pos;

else

cout<<endl<<ele<<"not fount";

getch();

}

**Name:-000**

**Roll No:-000**

**Assignment No:-**

**Assignment Title:-Implementation of program based on Graphs (Depth First Traversal)**

#include<iostream.h>

#include<conio.h>

class GRAPH

{

int n,u,VISITED[10];

int G[10][10];

public:

GRAPH(int);

void READ\_GRAPH();

void SHOW\_GRAPH();

void DFS(int);

};

GRAPH::GRAPH(int para)

{

n=para;

for(int i=1;i<=n;i++)

VISITED[i]=0;

}

void GRAPH::READ\_GRAPH()

{

cout<<endl<<"Enter Adjecency Matrix: \n";

for(int i=1;i<=n;i++)

for(int j=1;j<=n;j++)

cin>>G[i][j];

}

void GRAPH::SHOW\_GRAPH()

{

cout<<endl<<"Adjecency Matrix is : \n";

for(int i=1;i<=n;i++)

{

cout<<endl;

for(int j=1;j<=n;j++)

cout<<G[i][j]<<" ";

}

}

void GRAPH::DFS(int v)

{

int STK[10], top=0;

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

int u = v;

VISITED[u] = 1;

do

{

cout<<u<<" ";

for(int i=1;i<=n;i++)

{

if(G[u][i] == 1 && VISITED[i] == 0)

{

top = top +1;

STK[top] = i;

VISITED[i] = 1;

}

}

if(top == 0)

break;

else

{

u = STK[top];

top = top - 1;

}

}while(1);

}

void main()

{

clrscr();

int v,n;

cout<<endl<<"Enter no of vertices: ";

cin>>n;

GRAPH obj(n);

obj.READ\_GRAPH();

obj.SHOW\_GRAPH();

cout<<endl<<"Enter souce vertex: ";

cin>>v;

obj.DFS(v);

getch();

Name:- 000

Roll No:- 000

Assignment No:-

Assignment Title:- Implementation of program based on Graphs (Breadth First Traversal)

#include<iostream.h>

#include<conio.h>

class GRAPH

{

int n;

int G[10][10];

public:

GRAPH(int);

void READ\_GRAPH();

void SHOW\_GRAPH();

void BFS(int);

};

GRAPH::GRAPH(int para)

{

n=para;

}

void GRAPH::READ\_GRAPH()

{

cout<<endl<<"Enter Adjecency Matrix: \n";

for(int i=1;i<=n;i++)

for(int j=1;j<=n;j++)

cin>>G[i][j];

}

void GRAPH::SHOW\_GRAPH()

{

cout<<endl<<"Adjecency Matrix is : \n";

for(int i=1;i<=n;i++)

{

cout<<endl;

for(int j=1;j<=n;j++)

cout<<G[i][j]<<" ";

}

}

void GRAPH::BFS(int v)

{

int u, VISITED[10],QUE[10],rear,front;

rear=front=0;

for(int i=1;i<=n;i++)

VISITED[i]=0;

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

VISITED[v]=1;

u = v;

do

{

cout<<u<<" ";

for(int i=1;i<=n;i++)

{

if(G[u][i] == 1 && VISITED[i] == 0)

{

if(front == 0)

front = 1;

rear = rear + 1;

QUE[rear] = i;

VISITED[i] = 1;

}

}

if(front == 0)

break;

else

{

u = QUE[front];

if(front == rear)

front = rear = 0;

else

front = front + 1;

}

}while(1);

}

void main()

{

clrscr();

int v,n;

cout<<endl<<"Enter no of vertices: ";

cin>>n;

GRAPH obj(n);

obj.READ\_GRAPH();

obj.SHOW\_GRAPH();

cout<<endl<<"Enter souce vertex: ";

cin>>v;

obj.BFS(v);

getch();

}

**Name:-000**

**Roll No:-000**

**Practical Name:-Implementation of program based on Graph-obtaining Shortest Path(using Dijkstra algorithm).**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

#include <iostream>

using namespace std;

int a[20][20], q[20], visited[20], n, i, j, f = 0, r = -1;

void bfs(int v)

{

    for (i = 1; i <= n; i++)

   {

    if (a[v][i] && !visited[i])

         q[++r] = i;

  }

    if (f <= r)

    {

            visited[q[f]] = 1;

             bfs(q[f++]);

        }

}

int main()

{

    int v;

    cout << "\n Enter the number of vertices: ";

    cin >> n;

     for (i = 1; i <= n; i++)

     {

        q[i] = 0;

        visited[i] = 0;

     }

    cout << "\n Enter graph data in matrix form:\n";

    for (i = 1; i <= n; i++)

    {

        for (j = 1; j <= n; j++)

         {

            cin >> a[i][j];

         }

    }

    cout << "\n Enter the starting vertex: ";

     cin >> v;

    bfs(v);

     cout << "\n The nodes that are reachable are:\n";

     for (i = 1; i <= n; i++)

     {

        if (visited[i])

        cout << i << "\t";

        else

        {

            cout << "\n Bfs is not possible";

             break;

        }

     }

    cout << endl;

    return 0;

}

Name:-000

Roll No:-000

Title:-Implementation of program based on Linear Search.

--------------------------------------------------------------------------------------------------------------------------------

#include "iostream.h"

#include "conio.h"

#include "stdlib.h"

class LIST

{

int \*A,size;

public:

LIST(int);

void SET\_LIST();

void VIEW\_LIST();

int LINEAR\_SEARCH(int);

};

LIST::LIST(int par)

{

size=par;

A =new int[size+1];

}

void LIST::SET\_LIST()

{

//cout<<endl<<"Enter list elements : ";

for(int i=1;i<=size;i++)

A[i]=random(1000);

}

void LIST::VIEW\_LIST()

{

cout<<"List elements are : ";

for(int i=1;i<=size;i++)

cout<<A[i]<<" ";

}

int LIST::LINEAR\_SEARCH(int x)

{

for(int i=1;i<=size;i++)

{

if(A[i]==x)

return i;

}

return 0;

}

void main()

{

int n,pos,ele;

clrscr();

cout<<"\n Enter size of array : ";

cin>>n;

LIST obj(n);

obj.SET\_LIST();

cout<<endl<<"List before sorting : \n";

obj.VIEW\_LIST();

cout<<"\n Enter element to search: ";

cin>>ele;

pos = obj.LINEAR\_SEARCH(ele);

if(pos !=0)

cout<<endl<<ele<<"fount at: "<<pos;

else

cout<<endl<<ele<<"Not Fount:";

getch();

}