Assignment No :- 1.1

Name:- Jayesh Vasant Falak

RollNo :- 57

Assignment Title :- Implementation of program based on the **Array**.

Code :-

#include<iostream.h>

#include<conio.h>

class ARRAY

{

private:

int A[7],size,n;

public:

ARRAY();

void ADD\_END(int ele);

int DEL\_END();

void LIST\_ALL();

};

void ARRAY::ARRAY()

{

size=6;

n=0;

}

void ARRAY::ADD\_END(int ele)

{

if(n==size)

{

cout<<endl<<"Array is full";

return;

}

n=n+1;

A[n]=ele;

}

int ARRAY::DEL\_END()

{

if(n==0)

{

cout<<endl<<"Array is empty";

return NULL;

}

int ele=A[n];

n=n-1;

return ele;

}

void ARRAY::LIST\_ALL()

{

if(n==0)

{

cout<<endl<<"Array is empty";

}

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

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

}

void MENU()

{

ARRAY obj;

do

{

int option,ele;

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

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

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

cout<<endl<<"3.LIST\_ALL";

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

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

cin>>option;

switch(option)

{

case 1:

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

cin>>ele;

obj.ADD\_END(ele);

break;

case 2:

obj.DEL\_END();

break;

case 3:

obj.LIST\_ALL();

break;

case 4:

cout<<endl<<"Exit";

return;

default:

cout<<endl<<"Invalid option";

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch(); }

Practical No:1.2

Practical Title:- Implementation of program based on **Matrix.**

Name:- Jayesh Vasant Falak

Roll No : 57

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

#include "iostream.h"

#include "conio.h"

class MATRIX

{

int r1,r2,r3,c1,c2,c3;

int A[5][5],B[5][5],C[5][5];

public:

void READ();

void SHOW();

void ADD();

void SUB();

void MUL();

};

void MATRIX::READ()

{

cout<<endl<<"enter the size of r1: ";

cin>>r1;

cout<<endl<<"enter the size of c1: ";

cin>>c1;

cout<<endl<<"enter the size of r2: ";

cin>>r2;

cout<<endl<<"enter the size of c2: ";

cin>>c2;

cout<<endl<<"enter the element of MATRIX 1: ";

for(int i=1;i<=r1;i++)

for(int j=1;j<=c1;j++)

cin>>A[i][j];

cout<<endl<<"enter the element of MATRIX 2 :";

for(i=1;i<=r2;i++)

for(j=1;j<=c2;j++)

cin>>B[i][j];

}

void MATRIX::SHOW()

{

cout<<endl<<"elements of MATRIX 1 are:"<<endl;

for(int i=1;i<=r1;i++)

{

for(int j=1;j<=c1;j++)

cout<<A[i][j]<<"\t";

cout<<"\n";

}

cout<<endl<<"elements of MATRIX 2 are :"<<endl;

for(i=1;i<=r2;i++)

{

for(int j=1;j<=c2;j++)

cout<<B[i][j]<<"\t";

cout<<"\n";

}

cout<<endl<<"elements of MATRIX 3 are :"<<endl;

for(int k=1;k<=r2;k++)

{

for(int j=1;j<=c2;j++)

cout<<C[k][j]<<"\t";

cout<<"\n";

}

}

void MATRIX::ADD()

{

if(r1==r2 && c1==c2)

{

for(int i=1;i<=r2;i++)

for(int j=1;j<=c2;j++)

C[i][j] = A[i][j] + B[i][j];

}

}

void MATRIX::SUB()

{

if(r1==r2 && c1==c2)

{

for(int i=1;i<=r2;i++)

for(int j=1;j<=c2;j++)

C[i][j] = A[i][j] - B[i][j];

}

}

void MATRIX::MUL()

{

if(c1==r2)

{

for(int i=1;i<=r2;i++)

for(int j=1;j<=c2;j++)

{

C[i][j]=0;

for(int k=1;k<=c1;k++)

C[i][j] += A[i][j] \* B[i][j];

}

}

}

void MENU()

{

MATRIX m;

m.READ();

int ch;

do

{

cout<<endl<<"1 ADD";

cout<<endl<<"2 Substracton";

cout<<endl<<"3 Multiplication";

cout<<endl<<"4 Exit";

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

cin>>ch;

switch(ch)

{

case 1:

m.ADD();

m.SHOW();

break;

case 2:

m.SUB();

m.SHOW();

break;

case 3:

m.MUL();

m.SHOW();

break;

case 4:

return;

default:

cout<<endl<<"enter valid option";

}

}while(ch!=4);

}

void main()

{

clrscr();

MENU();

getch(); }

Practical No:1.3.1

Practical Title:- Implementation of program based **stack** (using array).

Name:- Jayesh Vasant Falak

Roll No : 57

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

#include "iostream.h"

#include "conio.h"

class STACK

{

// data

int A[100],size, top;

public:

// operations

STACK(int);

void PUSH(int);

int POP();

int PEEP();

void VIEW\_STACK();

};

STACK::STACK(int par)

{

// def of function

size = par;

top = 0;

}

void STACK::PUSH(int ele)

{

// def of function

if(top == size)

{

cout<<endl<<"Stack is full";

return;

}

top = top+1;

A[top] = ele;

}

int STACK::POP()

{

// def of function

if(top == 0)

{

cout<<endl<<"Stack is empty";

return NULL;

}

int ele = A[top];

top = top-1;

return ele;

}

int STACK::PEEP()

{

// def of function

if(top == 0)

{

cout<<endl<<"Stack is empty";

return NULL;

}

return A[top];

}

void STACK::VIEW\_STACK()

{

// def of function

if(top == 0)

{

cout<<endl<<"Stack is empty";

return;

}

cout<<endl<<"Stack elemets are : ";

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

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

}

void MENU()

{

int ele,n,opt;

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

cin>>n;

STACK obj(n);

do

{

cout<<endl<<"1 PUSH";

cout<<endl<<"2 POP";

cout<<endl<<"3 PEEP";

cout<<endl<<"4 VIEW\_STACK";

cout<<endl<<"5 EXIT MENU";

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

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

cin>>opt;

switch(opt)

{

case 1:

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

cin>>ele;

obj.PUSH(ele);

break;

case 2:

ele = obj.POP();

if(ele)

cout<<endl<<"Poped ele = "<<ele;

break;

case 3:

ele = obj.PEEP();

if(ele)

cout<<endl<<"The top element of stack is "<<ele;

break;

case 4:

obj.VIEW\_STACK();

break;

case 5:

return;

default:

cout<<endl<<"invalid input";

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch();

}

Practical No:1.3.2

Practical Title:- Implementation of program based on **Stack** (Using Linked List.)

Name:- Jayesh Vasant Falak

Roll No : 57

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

#include"iostream.h" #include "iostream.h"

#include "conio.h"

#include "stdlib.h"

class NODE

{

public:

int data;

NODE \*link;

};

class STACK

{

NODE \*top;

public:

STACK();

void PUSH(int);

int POP();

void PEEP();

void VIEW();

int IS\_EMPTY();

};

STACK::STACK()

{

top=NULL;

}

void STACK::PUSH(int ele)

{

NODE \*NN;

NN=new NODE();

if(NN == NULL)

{

cout<<endl<<"LIST IS FULL";

return;

}

NN->data=ele;

NN->link=NULL;

if(top==NULL)

top=NN;

else

{

NN->link=top;

top=NN;

}

}

int STACK::POP()

{

if(top == NULL)

{

cout<<endl<<"LIST IS EMPTY";

return NULL;

}

int ele=top->data;

NODE \*TEMP=top;

top=top->link;

delete TEMP;

return ele;

}

int STACK::IS\_EMPTY()

{

if(top == NULL)

return 1;

else

return 0;

}

void STACK::PEEP()

{

cout<<top->data;

}

void STACK::VIEW()

{

if(top==NULL)

{

cout<<endl<<"LIST IS EMPTY";

return;

}

NODE \*ptr=top;

cout<<endl<<"LIST Elements Are :";

while(ptr!=NULL)

{

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

ptr=ptr->link;

}

}

void MENU()

{

int ele,opt;

STACK obj;

do

{

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

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

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

cout<<endl<<"3 Return TopMost element";

cout<<endl<<"4 View the element";

cout<<endl<<"5 EXIT";

cout<<endl<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

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

cin>>opt;

switch(opt)

{

case 1:

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

cin>>ele;

obj.PUSH(ele);

obj.VIEW();

break;

case 2:

cout<<endl<<"Deleted element:";

obj.POP();

obj.VIEW();

break;

case 3:

obj.PEEP();

break;

case 4:

obj.VIEW();

break;

case 5:

return;

default:

cout<<endl<<"Invalid Element";

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch();

}

Practical No:1.4.1

Practical Title:- Implementation of program based on **Queue** (using array).

Name:- Jayesh Vasant Falak

Roll No : 57

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

#include "iostream.h"

#include "conio.h"

class QUEUE

{

int A[50],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<<"Que is full";

else

{

if(front==0)

front=1;

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) // only 1 ele in Que

rear =front = 0;

else

front = front + 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,ele,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();

}

Practical No:1.4.2

Practical Title:- Implementation of program based on **Circular Queue** (using array).

Name:- Jayesh Vasant Falak

Roll No : 57

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

#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();

}

Practical No:1.4.3

Practical Title:- Implementation of program based on **Queue** (using linked list).

Name:- Jayesh Vasant Falak

Roll No : 57

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

#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();

}

Practical No:1.4.4

Practical Title:- Implementation of program based on **Priority Queue** .

Name:- Jayesh Vasant Falak

Roll No : 57

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

#include”iostream.h”

#include”conio.h”

Class PQUEUE

{

// private:

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();

}

Practical No:1.5

Practical Title:- Implementation of program based on **DeQueue**.

Name:- Jayesh Vasant Falak

Roll No : 57

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

#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

{

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();

}

Practical No:1.6.1

Practical Title:- Implementation of program based on **Linear Linked List(singly).**

Name:- Jayesh Vasant Falak

Roll No : 57

#include "iostream.h"

#include "conio.h"

class NODE

{

public:

int data;

NODE \*link;

};

class LIST

{

// data

NODE \*start;

public:

// operations

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()

{

// def of function

start = 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->link = NULL;

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

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

start=NN;

else //case - III Not Full - Not Empty

{

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

start = NN;

}

}

/////////////////////////////////////////////////////////////////

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->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)

{

// 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->link = NULL;

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

if(pos == 1)

{

NN->link = start;

start = NN;

}

else

{

int count = 1;

// ---- using one pointer variable ----

/\*

NODE \*ptr=start;

while(count < pos-1)

{

ptr = ptr->link;

count = count + 1;

}

NN->link = ptr->link;

ptr->link = NN;

\*/

// ---- 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()

{

// def of function

if(start == NULL)

return 1;

else

return 0;

}

/////////////////////////////////////////////////////////////////

void LIST::VIEW()

{

// def of function

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<<"Deleted element = "<<ele;

obj.VIEW();

}

break;

case 5:

if(!obj.IS\_EMPTY())

{

ele = obj.DEL\_LAST();

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

obj.VIEW();

}

break;

case 6:

if(!obj.IS\_EMPTY())

{

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

cin>>pos;

ele = obj.DEL\_POS(pos);

cout<<endl<<"Deleted 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();

}

/////////////////////////////////////////////////////////////////

Practical No:1.6.2

Practical Title:- Implementation of program based on **Circular Linked List**.

Name:- Jayesh Vasant Falak

Roll No : 57

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

#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);

void ADD\_LAST(int);

int DEL\_FIRST();

int DEL\_LAST();

void VIEW\_FWD();

int IS\_EMPTY();

};

/////////////////////////////////////////////////////////////////

LIST::LIST()

{

start = NULL;

end = 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;

if(start==NULL)

{

start = NN;

end = NN;

end->next = NN;

}

else

{

NN->next = start; start = NN;

end->next = NN; //start

}

}

/////////////////////////////////////////////////////////////////

void LIST::ADD\_LAST(int ele)

{

NODE \*NN;

NN= new NODE();

if(NN == NULL)

{

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

return;

}

NN->data = ele;

NN->next = NULL;

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()

{

if(start == NULL)

{

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

return NULL;

}

else

{

int ele = start->data;

NODE \* TEMP = start;

if(start->next == start) {

start = NULL;

end = NULL;

}

else

{

start = start->next;

end->next = start;

}

delete TEMP;

return ele;

}

}

/////////////////////////////////////////////////////////////////

int LIST::DEL\_LAST()

{

if(start == NULL)

{

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

return NULL;

}

else

{

int ele = end->data;

NODE \* TEMP = end;

if(start->next == start)

{

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()

{

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();

}

Practical No:1.6.3

Practical Title:- Implementation of program based on **Doubly Linked List**.

Name:- Jayesh Vasant Falak

Roll No : 57

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

#include"iostream.h"

#include"conio.h"

class NODE

{

public:

int data;

NODE \*next, \*prev;

};

class LIST

{

NODE \*start;

public:

LIST();

void ADD\_FIRST(int);

void ADD\_LAST(int);

int DEL\_FIRST();

int DEL\_LAST();

void VIEW\_FWD();

void VIEW\_BACK();

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;

if(start == NULL)

start = NN;

else

{

NN->next = start;

start->prev = NN;

start = NN;

}

}

///////////////////////////////

void LIST::ADD\_LAST(int ele)

{

NODE \*NN;

NN = new NODE();

NN->data = ele;

NN->next = NULL;

NN->prev = NULL;

if(start == NULL)

start = NN;

else

{

NODE \*ptr;

ptr= start;

while(ptr->next != NULL)

{

ptr= ptr->next;

}

ptr->next = NN;

NN->prev = ptr;

}

}

int LIST::DEL\_FIRST()

{

if(start == NULL)

{

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

return NULL;

}

else

{

int ele = start->data;

NODE \* TEMP = start;

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)

start = NULL;

else

ptr2->next = NULL;

delete TEMP;

return ele;

}

void LIST::VIEW\_FWD()

{

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;

while(ptr->next != NULL)

{

ptr=ptr->next;

}

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

while(ptr != NULL)

{

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

ptr=ptr->prev;

}

}

int LIST::IS\_EMPTY()

{

if(start == NULL)

return 1;

else

return 0;

}

void MENU()

{

int ele,opt;

LIST obj;

do

{

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

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

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

cout<<endl<<"4 DEL form Last";

cout<<endl<<"5 VIEW from Forward Position";

cout<<endl<<"6 VIEW from Backward Postion";

cout<<endl<<"7 EXIT MENU";

cout<<endl<<"=============================";

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

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<<"Deleted element = "<<ele;

obj.VIEW\_FWD();

}

break;

case 4:

if(!obj.IS\_EMPTY())

{

ele = obj.DEL\_LAST();

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

obj.VIEW\_FWD();

}

break;

case 5:

obj.VIEW\_FWD();

break;

case 6:

obj.VIEW\_BACK();

break;

case 7:

return;

default:

cout<<endl<<"Invalid Chioce";

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch();

}

Practical No:1.7.1

Practical Title:- Implementation of program based on **Polynomial Addition / Subtraction** (Using Array

Name:- Jayesh Vasant Falak

Roll No : 57

// 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.READ\_POLYEXPR2();

//obj.ADD\_POLYEXPR();

//obj.SUB\_POLYEXPR();

obj.VIEW\_POLYEXPR();

getch();

}

Practical No:1.7.2

Practical Title:- Implementation of program based on **Polynomial Addition / Subtraction** (Using Linked List.

Name:- Jayesh Vasant Falak

Roll No : 57

---------------------------------------------------------------------------------------------------------------- // 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();

}

Practical No:2.1

Practical Title:- Implementation of program based on **Binary Search Tree.**

Name:- Jayesh Vasant Falak

Roll No : 57

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

#include "iostream.h"

#include "conio.h"

class NODE

{

public:

int data;

NODE \*left, \*right;

};

class TREE

{

// data

public:

NODE \*root;

// operations

TREE();

void ADD(int); // 1

void DEL(int); // 2

void PRE\_ORD(NODE \*); // 3

void IN\_ORD(NODE \*); // 4

void POST\_ORD(NODE \*);// 5

NODE \* FIND\_IIO\_SUCCESSOR(NODE \*);

void MENU();

};

/////////////////////////////////////////////////////////////////

TREE::TREE()

{

// def of function

root = NULL;

}

/////////////////////////////////////////////////////////////////

void TREE::ADD(int ele)

{

// def of function

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

NODE \*NN;

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

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

NN->data = ele;

NN->left = NULL;

NN->right = NULL;

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

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

root=NN;

else //case - III Not Full - Not Empty

{

NODE \*par = NULL;

NODE \*ptr = root;

while(ptr != NULL)

{

par = ptr;

if(ele < ptr->data)

ptr = ptr->left;

else

ptr = ptr->right;

}

if(ele < par->data)

par->left = NN;

else

par->right = NN;

}

}

/////////////////////////////////////////////////////////////////

NODE \* TREE:: FIND\_IIO\_SUCCESSOR(NODE \*ptr)

{

NODE \*par\_ssr = ptr;

NODE \*ssr = ptr->right;

while(ssr->left != NULL)

{

par\_ssr = ssr;

ssr = ssr->left;

}

// delete ssr

if(par\_ssr == ptr)

par\_ssr->right = ssr->right;

else

par\_ssr->left = ssr->right;

return ssr;

}

/////////////////////////////////////////////////////////////////

void TREE::DEL(int ele)

{

if(root == NULL)

cout<<endl<<"Tre is empty";

else // Tree not empty

{

NODE \*ptr=root;

NODE \*par=NULL;

// find the node to be deleted with his parent

while(ptr!=NULL)

{

if(ptr->data==ele)

break; // node found

else

{

par = ptr;

if(ele<ptr->data)

ptr=ptr->left;

else

ptr=ptr->right;

}

}

if(ptr == NULL) // node not found

cout<<"Element Not Found";

else // node found

{

NODE \*TEMP=ptr;

if(ptr->left==NULL && ptr->right==NULL) // zero child

{

if(par == NULL) // ptr is root of tree

root = NULL;

else

if(ele<par->data)

par->left=NULL;

else

par->right=NULL;

}

else

{

if(ptr->left == NULL || ptr->right == NULL) // 1 child

{

// find out child

NODE \*ch;

if(ptr->left==NULL)

ch = ptr->right;

else

ch=ptr->left;

// set links

if(par == NULL) // ptr is root of tree

root = ch;

else

{

if(ele<par->data)

par->left=ch;

else

par->right=ch;

}

}

else // 2 children

{

NODE \*IIOS = FIND\_IIO\_SUCCESSOR(ptr);

IIOS->left = ptr->left;

IIOS->right = ptr->right;

if( ele < par->data )

par->left = IIOS;

else

par->right = IIOS;

}

}

delete TEMP;

}

}

}

/////////////////////////////////////////////////////////////////

void TREE::PRE\_ORD( NODE \*ptr)

{

// def of function

if(ptr != NULL)

{

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

PRE\_ORD(ptr->left);

PRE\_ORD(ptr->right);

}

}

/////////////////////////////////////////////////////////////////

void TREE::IN\_ORD( NODE \*ptr)

{

// def of function

if(ptr != NULL)

{

IN\_ORD(ptr->left);

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

IN\_ORD(ptr->right);

}

}

/////////////////////////////////////////////////////////////////

void TREE::POST\_ORD( NODE \*ptr)

{

// def of function

if(ptr != NULL)

{

POST\_ORD(ptr->left);

POST\_ORD(ptr->right);

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

}

}

/////////////////////////////////////////////////////////////////

void TREE::MENU()

{

int ele, opt;

do

{

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

cout<<endl<<"1 Add Node";

cout<<endl<<"2 Delete Node";

cout<<endl<<"3 Pre-Order Traversal";

cout<<endl<<"4 In-Order Traversal";

cout<<endl<<"5 Post-Order Traversal";

cout<<endl<<"6 Exit";

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

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

cin>>opt;

switch(opt)

{

case 1:

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

cin>>ele;

ADD(ele);

IN\_ORD(root);

break;

case 2:

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

cin>>ele;

DEL(ele);

if(root != NULL)

IN\_ORD(root);

else

cout<<endl<<"Tree empty";

break;

case 3:

if(root != NULL)

PRE\_ORD(root);

else

cout<<endl<<"Tree empty";

break;

case 4:

if(root != NULL)

IN\_ORD(root);

else

cout<<endl<<"Tree empty";

break;

case 5:

if(root != NULL)

POST\_ORD(root);

else

cout<<endl<<"Tree empty";

break;

case 6:

return;

default:

cout<<endl<<"Invalid input";

}

}while(1);

}

/////////////////////////////////////////////////////////////////

void main()

{

TREE obj;

clrscr();

obj.MENU();

getch();

}

/////////////////////////////////////////////////////////////////

Practical No:3.1

Practical Title:- Implementation of program based on **Graph – Depth First Traversal**.

Name:- Jayesh Vasant Falak

Roll No : 57

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

#include"iostream.h"

#include"conio.h"

class GRAPH

{

int n;

int G[10][10];

int u,VISITED[10];

public:

GRAPH(int);

void READ\_GRAPH();

void SHOW\_GRAPH();

void DFS\_NR(int);

};

GRAPH::GRAPH(int para)

{

n=para;

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

VISITED[i]=0;

}

void GRAPH::READ\_GRAPH()

{

cout<<endl<<"Enter Adjencency matrix: ";

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

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

cin>>G[i][j];

}

void GRAPH::SHOW\_GRAPH()

{

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

{

cout<<endl;

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

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

}

}

void GRAPH::DFS\_NR(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 n,v;

cout<<endl<<"Enter the number of vertices: ";

cin>>n;

GRAPH obj(n);

obj.READ\_GRAPH();

obj.SHOW\_GRAPH();

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

cin>>v;

obj.DFS\_NR(v);

getch();

}

Practical No:3.2

Practical Title:- Implementation of program based on **Graph – Breadth First Traversal.**

Name:- Jayesh Vasant Falak

Roll No : 57

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

#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 Adjencency matrix: ";

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

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

cin>>G[i][j];

}

void GRAPH::SHOW\_GRAPH()

{

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(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 n,v;

cout<<endl<<"Enter the number of vertices: ";

cin>>n;

GRAPH obj(n);

obj.READ\_GRAPH();

obj.SHOW\_GRAPH();

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

cin>>v;

obj.BFS(v);

getch();

}

Practical No:-4.2.1

Practical Title:- Implementation of program based on **Linear Search** (using array).

Name:- Jayesh Vasant Falak

Roll No : 57

#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 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,ele,pos;

clrscr();

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

cin>>n;

LIST obj(n);

obj.SET\_LIST();

cout<<endl<<"List of elements: \n";

obj.VIEW\_LIST();

cout<<endl<<"Enter ele for search: ";

cin>>ele;

pos= obj.LINEAR\_SEARCH(ele);

if(pos != 0)

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

else

cout<<endl<<ele<<"Not found";

getch();

}

Practical No:-4.2.2

Practical Title:- Implementation of program based on **Binary Search** (using array )

Name:- Jayesh Vasant Falak

Roll No : 57

#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();

};

LIST::LIST(int par)

{

size=par;

A=new int[size+1];

}

void LIST::SET\_LIST()

{

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

A[i]=random(1000);

}

void main()

{

int n,pos,ele;

clrscr();

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

cin>>n;

LIST obj(n);

obj.SET\_LIST();

cout<<endl<<"LIST of elements :" ;

obj.VIEW\_LIST();

cout<<endl<<"list of elements: ";

cin>>ele;

pos = obj.BINARY\_SEARCH(ele);

if(pos !=0)

cout<<endl<ele<<"Found On "<<pos;

else

cout<<endl<<"NOt Found" ;

getch(); }

Practical No:-4.3

Practical Title:- Implementation of program for sorting techniques – **Bubble Sort.**

Name:- Jayesh Vasant Falak

Roll No : 57

#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();

};

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;

}

}

////////////////////////////////////////////////////

void main()

{

int n;

clrscr();

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

cin>>n;

LIST obj(n);

obj.SET\_LIST();

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

obj.VIEW\_LIST();

obj.BUBBLE\_SORT();

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

obj.VIEW\_LIST();

getch();

}

Practical No:-4.4

Practical Title:- Implementation of program for sorting techniques – **Selection Sort.**

Name:- Jayesh Vasant Falak

Roll No : 57

#include<iostream.h>

#include<conio.h>

#include<stdio.h>

#include<stdlib.h>

class LIST

{

int \*A,size;

public:

LIST(int);

void SET\_LIST();

void VIEW\_LIST();

int MIN(int);

void SELECTION\_SORT();

void SWAP(int,int);

};

LIST::LIST(int par)

{

size=par;

A=new int[size+1];

}

void LIST::SET\_LIST()

{

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

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

A[i]=random(1000);

}

void LIST::VIEW\_LIST()

{

cout<<"List element are:";

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

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

}

int LIST::MIN(int i)

{

int min=A[i], pos=i;

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

{

if(A[j]<min)

{

min=A[j];

pos=j;

}

}

return pos;

}

void LIST::SWAP(int i, int j)

{

int temp= A[i];

A[i]=A[j];

A[j]=temp;

}

void LIST::SELECTION\_SORT()

{

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

{

int min\_pos = MIN();

SWAP(i,min\_pos);

}

}

void main()

{

int n;

clrscr();

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

cin>>n;

LIST obj(n);

obj.SET\_LIST();

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

obj.VIEW\_LIST();

obj.SELECTION\_SORT();

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

obj.VIEW\_LIST();

getch();

}

Practical No:-4.5

Practical Title:- Implementation of program for sorting techniques – **Insertion Sort.**

Name:- Jayesh Vasant Falak

Roll No : 57

#include "iostream.h"

#include "conio.h"

#include "stdlib.h"

class LIST

{

// data

int \*A,size; // 2 sep arry

public:

// operations

LIST(int);

void READ\_ELE();

void INS\_SORT();

void LIST\_ALL();

};

LIST::LIST(int par)

{

// def of function

size = par;

A = new int[size+1];

}

void LIST::READ\_ELE()

{

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

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

//cin>>A[i];

A[i]=random(1000);

}

void LIST::INS\_SORT()

{

for(int i=2;i<=size ;i++)

{

int ele = A[i];

int j= i-1;

while( j>=1 && ele>A[j] )

{

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

j=j-1;

}

A[j+1]=ele;

}

}

void LIST::LIST\_ALL()

{

// def of function

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

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

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

}

///////////////////////////////////////////////////////

void main()

{

int n;

clrscr();

cout<<endl<<"Enter size of arry : ";

cin>>n;

LIST obj(n);

/////////////////////////////////////////

obj.READ\_ELE();

cout<<endl<<"Elements before sorting";

obj. LIST\_ALL();

obj.INS\_SORT();

cout<<endl<<"Elements after sorting";

obj.LIST\_ALL();

getch();

}

Practical No:-4.6

Practical Title:- Implementation of program for sorting techniques – **Radix Sort.**

Name:- Jayesh Vasant Falak

Roll No : 57

#include<iostream.h>

#include<conio.h>

#include<stdio.h>

#include<stdlib.h>

class LIST

{

int \*A,size;

public:

LIST(int);

void SET\_LIST();

void VIEW\_LIST();

void RADIX\_SORT();

};

LIST::LIST(int par)

{

size=par;

A=new int[size+1];

}

void LIST::SET\_LIST()

{

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

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

A[i]=random(1000);

}

void LIST::VIEW\_LIST()

{

cout<<"List element are:";

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

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

}

void LIST::RADIX\_SORT()

{

//-------------declare queue(s)--------------

//int F[100],R[100],QUE[10][100];

//or

int \*\*QUE,\*F,\*R;

QUE = new int\* [10];

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

QUE[i]=new int[size];

F =new int[10];

R=new int[10];

//----------sort the list now--------

int d=1;

for(i=1;i<=5;i++)

{ //-------initializing queue(s)---------

for(int l=0;l<=9;l++)

{

F[l]=0;

R[l]=0;

}

//--------add all element in queues--------

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

{

int k=(A[j]/d)%10;

if(F[k]==0)

F[k]=1;

R[k]=R[k]+1;

QUE[k][R[k]]=A[j];

}

//----collect all element from queue(s)-----

j=1;

for(int qn=0;qn<=9;qn++)

{

if(F[qn] != 0) //Que not empty

for(int m=F[qn];m<=R[qn];m++)

{

A[j]=QUE[qn][m];

j=j+1;

}

}d=d\*10;

}

}

void main()

{

int n;

clrscr();

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

cin>>n;

LIST obj(n);

obj.SET\_LIST();

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

obj.VIEW\_LIST();

obj.RADIX\_SORT();

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

obj.VIEW\_LIST();

getch();

}

Practical No:-4.7

Practical Title:- Implementation of program for sorting techniques – **Quick Sort.**

Name:- Jayesh Vasant Falak

Roll No : 57

#include "iostream.h"

#include "conio.h"

#include "stdlib.h"

class LIST

{

int \*A,size;

public:

LIST(int);

void SET\_LIST();

void VIEW\_LIST();

void QUICK\_SORT(int,int);

void PARTITION(int,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);

A[i+1]=9999; // +infinity

}

void LIST::VIEW\_LIST()

{

cout<<"List elements are : ";

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

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

}

void LIST::PARTITION(int m,int & p) // in-out type 'p'

{

int v=A[m]; int i=m;

do

{

do

{

i=i+1;

}while(A[i]<v);

do

{

p=p-1;

}while(A[p]>v);

if(i<p)

{

int temp=A[i];

A[i]=A[p];

A[p]=temp;

}

else

break;

}while(1);

A[m]=A[p];

A[p]=v;

}

void LIST::QUICK\_SORT(int p,int q)

{

if(p<q)

{

int j=q+1;

PARTITION(p,j);

QUICK\_SORT(p,j-1);

QUICK\_SORT(j+1,q);

}

}

////////////////////////////////////////////////////

void main()

{

int n;

clrscr();

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

cin>>n;

LIST obj(n);

obj.SET\_LIST();

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

obj.VIEW\_LIST();

obj.QUICK\_SORT(1,n);

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

obj.VIEW\_LIST();

getch();

}

Practical No:-4.8

Practical Title:- Implementation of program for sorting techniques – **Merge Sort.**

Name:- Jayesh Vasant Falak

Roll No : 57

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

#include "iostream.h"

#include "conio.h"

#include "stdlib.h"

class LIST

{

int \*A,size;

public:

LIST(int);

void SET\_LIST();

void VIEW\_LIST();

void MERGE\_SORT(int,int);

void MERGE(int,int,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::MERGE(int low,int mid,int high)

{

int \*B=new int[size+1];

int h=low,j=mid+1; // pointer for both sublist(l1 & l2)

int i=low; // pointer for auxilary list (l3)

while(h<=mid && j<=high)

{

if(A[h]<A[j])

{

B[i]=A[h];

h=h+1;

i=i+1;

}

else

{

B[i]=A[j];

j=j+1;

i=i+1;

}

}

if(h>mid) // l1 exhausted

{

// copy remaining l2 in B[]

for(int k=j;k<=high;k++)

{

B[i]=A[k];

i=i+1;

}

}

else // l2 exhausted

{

// copy remaining l1 in B[]

for(int k=h;k<=mid;k++)

{

B[i]=A[k];

i=i+1;

}

}

for(int k=low; k<=high; k++)

{

A[k]=B[k];

}

delete B;

}

void LIST::MERGE\_SORT(int low,int high)

{

if(low<high)

{

int mid=(low+high)/2;

MERGE\_SORT(low,mid);

MERGE\_SORT(mid+1,high);

MERGE(low,mid,high);

}

}

////////////////////////////////////////////////////

void main()

{

int n;

clrscr();

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

cin>>n;

LIST obj(n);

obj.SET\_LIST();

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

obj.VIEW\_LIST();

obj.MERGE\_SORT(1,n);

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

obj.VIEW\_LIST();

getch();

}

Practical No:-4.9

Practical Title:- Implementation of program for sorting techniques – **Heap Sort.**

Name:- Jayesh Vasant Falak

Roll No : 57

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

#include "iostream.h"

#include "conio.h"

#include "stdlib.h"

class LIST

{

int \*A,size;

public:

LIST(int);

void SET\_LIST();

void VIEW\_LIST();

void HEAP\_SORT();

void HEAPIFY();

void ADJUST(int,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);

cin>>A[i];

}

void LIST::VIEW\_LIST()

{

cout<<"List elements are : ";

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

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

}

void LIST::HEAP\_SORT()

{

// ---- step 1 ------

HEAPIFY();

// ---- step 1 ------

for(int i=size;i>1;i--)

{

// EXCHANAGE ELEMENTS--------

int temp = A[1];

A[1] = A[i];

A[i] = temp;

//or EXCH(1,i);

ADJUST(1,i-1);//ADJUST(int i,int n)

}

}

void LIST::HEAPIFY()

{

for(int i=size/2;i>=1;i--)

ADJUST(i,size);

}

void LIST::ADJUST(int i,int n)

{

int item = A[i]; int j = 2\*i;

while(j<=n)

{

if(j<n && A[j] < A[j+1])

j = j+1;

if(item >= A[j])

break;

else

A[j/2] = A[j];

j = j\*2;

}

A[j/2] = item;

}

////////////////////////////////////////////////////

void main()

{

int n;

clrscr();

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

cin>>n;

LIST obj(n);

obj.SET\_LIST();

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

obj.VIEW\_LIST();

obj.HEAP\_SORT();

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

obj.VIEW\_LIST();

getch();

}