#All Pair Shortest.cpp

#include<iostream.h>

#include<conio.h>

class path

{

int a[5][5],i,j,k,n,s,d;

public:

void insert();

void display();

};

void path::insert()

{

cout<<"\nEnter the no. of vertices";

cin>>n;

cout<<"\nEnter the matrix:";

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

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

{

cin>>a[i][j];

if(a[i][j]==-1)

a[i][j]=9999;

}

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

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

for(k=1;k<=n;k++)

if(a[i][j]<(a[i][k]+a[k][j]))

a[i][j]=a[i][j];

else

a[i][j]=(a[i][k]+a[k][j]);

}

void path::display()

{

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

{

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

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

cout<<"\n";

}

cout<<"\nEnter the source vertex:";

cin>>s;

cout<<"\nEnter the destination vertex:";

cin>>d;

cout<<"\nPath from Source "<<s<<" to destination "<<d<<" is ";

cout<<a[s][d];

}

void main()

{

clrscr();

path p;

p.insert();

cout<<"\n Shortest path is \n";

p.display();

getch();

}

#Array.cpp

#include<iostream.h>

#include<conio.h>

#include<process.h>

class demo

{

int a[10],i,j,k,n,item;

public:

void get();

void insert();

void del();

void dis();

};

void demo::get()

{

cout<<"\nEnter n";

cin>>n;

cout<<"\nEnter array element";

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

cin>>a[i];

}

void demo::insert()

{

cout<<"\nEnter position:";

cin>>k;

cout<<"Enter item:";

cin>>item;

j=n;

while(j>=k)

{

a[j+1]=a[j];

j--;

}

a[k]=item;

n++;

}

void demo::del()

{

cout<<"\nEnter position";

cin>>k;

j=k;

while(j<=n-1)

{

a[j]=a[j+1];

j++;

}

n--;

}

void demo::dis()

{

cout<<"\n Element are\n";

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

cout<<a[i]<<"\t";}

void main() {

clrscr(); demo d; int ch; d.get();

cout<<"\n1.Insert 2.del 3.dis 4.Exit\n";

while(ch!=4) {

cout<<"\nEnter choice"; cin>>ch;

switch(ch) {

case 1: d.insert();break;

case 2: d.del();break;

case 3: d.dis();break;

case 4: exit(0);

}

}

getch(); }

#BFS.cpp

#include<iostream.h>

#include<conio.h>

class bfstree

{

int reach[20],a[20][20],q[20],n,i,j,f,r,index;

public:

bfstree()

{

f=r=0;

index=1;

}

void get();

void bfs();

};

void bfstree::get()

{

cout<<"\nEnter number of vertices:";

cin>>n;

cout<<"\nEnter Adjacency matrix:";

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

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

{

reach[i]=0;

cin>>a[i][j];

}

}

void bfstree::bfs()

{

reach[1]=1;

f++;

r++;

q[r]=index;

cout<<"\nBFS is ";

while(f<=r)

{

index=q[f];

f++;

cout<<index<<"\t";

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

{

if(a[index][j]==1 && reach[j]!=1)

{

reach[j]=1;

r++;

q[r]=j;

}

}

}

}

void main()

{

clrscr();

bfstree b;

b.get();

b.bfs();

getch();

}

#Bubble\_sort.cpp

#include<iostream.h>

#include<conio.h>

class demo {

int a[10],temp;

int i,last,exch,j,n,temp1;

public:

void get();

void asc\_sort();

void dec\_sort();

void disp();

};

void demo::get()

{

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

cin>>n;

cout<<"\nEnter the array element:";

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

cin>>a[i];

}

void demo::asc\_sort() {

last=n;

for(i=1;i<=n-1;i++) {

exch=0;

for(j=1;j<=last-1;j++) {

if(a[j]>a[j+1]) {

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

exch=exch+1;

}

}

if(exch==0)

return;

else

last=last-1;

}

void demo::dec\_sort() {

last=n; for(i=1;i<=n-1;i++) {

exch=0;

for(j=1;j<=last-1;j++) {

if(a[j]<a[j+1]) {

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp; }

exch=exch+1; } }

if(exch==0)

return;

else

last=last-1; }

void demo::disp() {

cout<<"\nThe array element are";

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

cout<<a[i]<<"\t"; }

void main() {

clrscr(); demo d; d.get(); d.disp(); d.asc\_sort();

cout<<"\nAfter Ascending Sort:";

d.disp();

d.dec\_sort();

cout<<"\nAfter Descending Sort:";

d.disp();

getch();

}

**#CIRCULAR\_QUEUE\_ARRAY.CPP**

#include<iostream.h>

#include<conio.h>

class queue {

int a[5],r,f;

public:

queue() {

f=r=-1;

} void push(); void pop(); void show();

};

void queue::push() {

int item;

if(f==0 &&r==4 || f==r+1) {

cout<<"\n Overflow";

}

Else {

if(r==4)

r=-1; r++;

cout<<"\nEnter item:";

cin>>item;

a[r]=item;

if(f==-1) {

f=0;

} } }

void queue::pop() {

if(f==-1) {

cout<<"\nUnderflow"; }

else {

cout<<"\Deleted element is:"<<a[f];

if(f==r) { f=-1; r=-1; }

Else { if(f==4) f=0;

Else

f++;

} } }

void queue::show() {

if(f==-1) {

cout<<"\nEmpty :";

}

else if(f<=r) {

for(int i=f;i<=r;i++) {

cout<<"\n"<<a[i];

}

} else {

for(int i=f;i<=4;i++) {

cout<<"\n"<<a[i];

}

for(int j=0;j<=r;j++) {

cout<<"\n"<<a[j];

} } }

void main(){

queue s;

int ch;

clrscr();

do {

cout<<"\n 1: Push 2: Pop 3:show 4:exit ";

cout<<"\nEnter choice";

cin>>ch;

switch(ch) {

case 1: s.push();break;

case 2: s.pop(); break;

case 3: s.show(); break;

default: cout<<"\n Wrong Choice";

} }

while(ch<=3);

}

#DFS.cpp

#include<iostream.h>

#include<conio.h>

class dfstree

{

int a[20][20], visited[20],n,i,j;

public:

void dfs(int);

void get();

};

void dfstree::get()

{

cout<<"\nEnter the number of node";

cin>>n;

for(i=0;i<n;i++)

visited[i]=0;

cout<<"\nEnter the adjancy matrix:";

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

cin>>a[i][j];

}

}

dfs(0);

}

void dfstree::dfs(int v)

{

int k;

visited[v]=1;

cout<<"DFS:\t"<<v+1;

for(k=1;k<n;k++)

if(a[v][k]==1)

if(visited[k]==0)

dfs(k);

}

void main()

{

clrscr();

dfstree d;

d.get();

getch();

}

#Heap\_Sort.cpp

#include<iostream.h>

#include<conio.h>

class heap

{

int n,a[10],q,i,j,k,key,temp;

public:

void get();

void create();

void sort();

void display();

};

void heap::get()

{

cout<<"\Enter range:";

cin>>n;

cout<<"\nEnter the elements\n";

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

cin>>a[i];

}

void heap::create()

{

for(q=2;q<=n;q++) {

i=q; key=a[q]; j=i/2;

while(i>1 && key >a[j]) //desce orderkey<a[j]

{

a[i]=a[j]; i=j; j=i/2; if(j<1)

j=1; }

a[i]=key;

} }

void heap::sort() {

create();

cout<<"\nMax Heap Tree";

display();

for(q=n;q>=2;q--) {

temp=a[1];

a[1]=a[q];

a[q]=temp;

i=1;

key=a[1];

j=2;

if(j+1<q)

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

j++;

while(j<=q-1 && a[j]>key) {

a[i]=a[j];

i=j;

j=i\*2;

if(j+1<q)

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

j++;

else

if(j>n)

j=n;

a[i]=key;

} } }

void heap::display() {

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

cout<<a[i]<<"\t";

}

void main() {

clrscr(); heap h;

h.get(); h.sort();

cout<<"\nSorted element:";

h.display();

getch(); }

#Heap\_Tree.cpp

#include<iostream.h>

#include<conio.h>

class heap

{

int n,a[10],q,i,j,k,key;

public:

void get();

void create();

void display();

};

void heap::get()

{

cout<<"\nEnter Range:";

cin>>n;

cout<<"\nEnter the element:";

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

cin>>a[i];

}

void heap::create()

{

for(q=2;q<=n;q++)

{

i=q;

key=a[q];

j=i/2;

while(i>1 && key>a[j]) //change Min heap while(i>1 && key<a[j])

{

a[i]=a[j];

i=j;

j=i/2;

if(j<1)

j=1;

}

a[i]=key;

}

}

void heap::display()

{

cout<<"\nHeap Tree:";

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

cout<<a[i]<<"\t";

}

void main()

{

clrscr();

heap h;

h.get();

h.create();

h.display();

getch();

}

#Insertion\_Sort.cpp

#include<iostream.h>

#include<conio.h>

#include<stdlib.h>

#include<math.h>

class insert

{

int n,a[10],temp,ptr,q,i,j,k,key;

public:

void get();

void sort();

void display();

};

void insert::get()

{

cout<<"\nEnter Range:";

cin>>n;

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

a[i]=random(1000);

cout<<"\nElements are :";

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

cout<<a[i]<<"\t";

}

void insert::sort()

{

a[0]=-9999;

for(i=2;i<=n;i++)

{

temp=a[i];

ptr=i-1;

while(temp<a[ptr])

{

a[ptr+1]=a[ptr];

ptr--;

}

a[ptr+1]=temp;

}

}

void insert::display()

{

cout<<"\nSorted Element using Insertion Sort:";

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

cout<<a[i]<<"\t";

}

void main()

{

clrscr();

insert h;

h.get();

h.sort();

h.display();

getch();

}

**#LINEAR\_BINARY\_SEARCH.CPP**

#include<iostream.h>

#include<conio.h>

#include<process.h>

class demo

{

int a[10],i,j,n,f,temp,ele,demo,mid,low,high;

public:

void get();

void sort();

void linear();

void binary();

void dis();

};

void demo::get()

{

cout<<"\n Enter n:";

cin>>n;

cout<<"\nEnter array Elements:";

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

cin>>a[i];

}

void demo::linear()

{

int ele;

cout<<"\nEnter the element to be search";

cin>>ele;

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

{

if(a[i]==ele)

{

cout<<"\nSuccessful search";

cout<<"\nElement is found at position "<<i;

return;

}

}

if(i>n)

{

cout<<"\nUnsuccessful search:";

cout<<"\nElement is not found ";

}

}

void demo::sort()

{

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

{

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

{

if(a[j]<a[j+1])

{

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

}

void demo::binary()

{

cout<<"\nEnter element to be search ";

cin>>ele;

f=0;

low=1;

high=n;

while(low<=high)

{

mid=(low+high)/2;

if(a[mid]==ele)

{

f=1;

cout<<"\nElement is found at :"<<mid;

return;

}

else if(a[mid]<ele)

low=mid+1;

else if(a[mid]>ele)

high=mid-1;

}

if(f==0)

cout<<"\n Element is not found:";

}

void demo::dis()

{

cout<<"\n Element are \n";

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

cout<<a[i]<<"\t";

}

void main()

{

clrscr();

demo d;

int ch;

d.get();

d.dis();

cout<<"\n 1:Linear 2:Binary 3:exit\n";

while(ch!=3)

{

cout<<"\nEnter Choice:";

cin>>ch;

switch(ch)

{

case 1: d.linear(); break;

case 2: d.sort();

d.dis();

d.binary(); break;

case 3: exit(0); break;

}

}

getch();

}

#Matrix.cpp

#include<iostream.h>

#include<conio.h>

class matrix {

int a[5][5],b[5][5],c[5][5],d[5][5];

int p,q,i,j,k,n,m;

public:

void get(); void add();

void sub(); void mul();

};

void matrix::get() {

cout<<"\n enter number of row and column:\n";

cin>>n>>m;

cout<<"\n enter the first matrix:\n";

for(i=0;i<n;i++) {

for(j=0;j<m;j++)

cin>>a[i][j];

}

cout<<"\n enter number of row and cloumn:\t";

cin>>p>>q;

cout<<"\n enter the second matrix:\n";

for(i=0;i<p;i++) {

for(j=0;j<q;j++)

cin>>b[i][j];

} }

void matrix::add() {

cout<<"\n the adition of two matrix is :\n";

for(i=0;i<n;i++) {

for(j=0;j<m;j++) {

c[i][j]=a[i][j]+b[i][j];

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

} cout<<"\n";

} }

void matrix::sub() {

cout<<"\n the subtraction of two matrix is:\n";

for (i=0;i<n;i++) {

for(j=0;j<m;j++) {

d[i][j]=a[i][j]-b[i][j];

cout<<d[i][j]<<"\n";

} cout<<"\n";

} }

void matrix::mul() {

cout<<"\n the matrix multiplication is :\n";

if(m==p) {

for(i=0;i<n;i++) {

for(j=0;j<q;i++) {

c[i][j]=0;

for(k=0;k<p;k++); {

c[i][j]=c[i][j]+a[i][k]\*b[k][j];

} }

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

}

cout<<"\n";

}

else

cout<<"\nMatrix multiplication not possible";

}

void main() {

clrscr();

matrix m;

m.get();

m.add();

m.mul();

getch();

}

#**Merge\_Sort.cpp**

#include<iostream.h>

#include<conio.h>

#include<stdio.h>

int n;

class merge {

int a[10],b[10],i,j;

public:

void read();

void merge\_sort(int l,int h);

void merge1(int l,int m, int h);

void disp();

};

void merge::read() {

for(i=0;i<n;i++)

cin>>a[i];

}

void merge::merge\_sort(int l,int h) {

int mid;

if(l<h) {

mid=int((l+h)/2);

merge\_sort(l,mid);

merge\_sort(mid+1,h);

merge1(l,mid,h);

} }

void merge::merge1(int low,int m,int high) {

int h=low;

int i=low;

j=m+1;

while((h<=m)&&(j<=high)) {

if(a[h]<=a[j]) //Change descending order if(a[h]>=a[j]) {

b[i]=a[h];

i++; h++;

}

Else {

b[i]=a[j];

i++; j++;

} }

if(h<=m) {

while(h<=m) {

b[i]=a[h];

i++; h++;

} }

Else {

while(j<=h) {

b[i]=a[j];

i++; j++;

} }

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

a[k]=b[k];

}

void merge::disp() {

for(i=0;i<n;i++)

cout<<a[i]<<"\t";

}

void main() {

clrscr(); int l,h; merge m;

cout<<"\nEnter Elements"; cin>>n;

h=n-1; l=0;

m.read();

cout<<"\n\nDisplay the array elements\n";

m.disp();

m.merge\_sort(l,h);

cout<<"\nAfter Sorting\n";

m.disp(); getch(); }

**#QUEUE\_USING\_ARRAY.CPP**

#include<iostream.h>

#include<conio.h>

#include<process.h>

int m;

class queue {

int f,r,q[10],n,i;

public:

queue() {

f=r=0; }

void insert(); void del(); void dis();

};

void queue::insert() {

if(r==m)

cout<<"\nOverflow";

else {

cout<<"\nEnter Element in Queue=";

cin>>n;

if(f==0)

f=1; r++;

q[r]=n;

} }

void queue::del() {

if(f==0)

{

cout<<"\nUnderflow";

}

Else {

int n;

n=q[f];

if(f==r)

f=r=0;

else

f++;

cout<<"\nDeleted element is "<<n;

}

}

void queue::dis()

{

if(f==0)

cout<<"\nUnderflow";

else {

cout<<"\nElements in queue are:";

for(i=f;i<=r;i++)

cout<<q[i]<<"\t";

}

}

void main() {

clrscr();

queue q;

int ch;

cout<<"Enter size of queue";

cin>>m;

cout<<"\n 1.insert 2.display 3.delete 4. exit \n";

while(ch!=4) {

cout<<"\nEnter ch:";

cin>>ch;

switch(ch) {

case 1: q.insert(); break;

case 2: q.dis(); break;

case 3: q.del(); break;

case 4:exit(0);

} }

getch();

}

**#QUEUE\_USING\_LINKED\_LIST.CPP**

#include<conio.h> #include<iostream.h>

#include<process.h>

class queue {

int info, ele,c;

queue \*node,\*link,\*start,\*move;

public:

queue() {

start=NULL; c=0;

}

void insert(); void del(); void dis();

};

void queue::insert() {

node=new queue;

if(c<3) {

cout<<"\nEnter Info:";

cin>>ele;

node->info=ele;

node->link=NULL;

if(start==NULL) {

start=node;

c++;

return;

}

Else {

move=start;

while(move->link!=NULL)

move=move->link;

move->link=node;

c++;

} }

else

cout<<"\n Overflow";

}

void queue::del() {

move=start;

if(move!=NULL) {

move=move->link;

cout<<"\nDeleted Element is:"<<start->info;

start=move;

}

else

cout<<"\nUnderflow:";

}

void queue::dis() {

move=start;

if(move==NULL) {

cout<<"\n Queue is empty";

return;

}

Else {

while(move!=NULL) {

cout<<move->info<<"\t";

move=move->link;

} } }

void main() {

clrscr(); int ch; queue s;

cout<<"\n1.Insert 2.Show 3.Delete 4.Exit";

while(ch!=4) {

cout<<"\nEnter Choice";

cin>>ch;

switch(ch) {

case 1:s.insert();break;

case 2:s.dis();break;

case 3:s.del();break; case 4:exit(0); } }

getch(); }

#Queqe\_Sort.cpp

#include<iostream.h>

#include<conio.h>

#include<string.h>

class demo {

int x[20],temp;

int a,n,i,j,left,right;

public:

void get();

void asort(int,int);

int partition(int,int);

void disp();

};

void demo::get() {

cout<<"\nEnter the array size:";

cin>>n;

cout<<"\nEnter the array element:";

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

cin>>x[i];

asort(1,n);

}

void demo::asort(int p,int q) {

if(p<q) {

j=partition(p,q);

asort(p,j-1);

asort(j+1,q);

}

}

int demo::partition(int lb, int ub) {

a=x[lb];

left=lb+1;

right=ub;

do //for Descending

{

while(x[left]<a) // while(x[left]>a)

left++;

while(x[right]>a) // while(x[right]<a)

right--;

if(left<right)

{

temp=x[left];

x[left]=x[right];

x[right]=temp;

}

}

while(left<=right);

x[lb]=x[right];

x[right]=a;

return(right);

}

void demo::disp()

{

cout<<"\nThe array element are:";

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

cout<<x[i]<<"\t";

}

void main()

{

clrscr();

demo d;

d.get();

cout<<"\nAfter Ascending sort";

d.disp();

getch();

}

#Selection\_Sort.cpp

#include<iostream.h>

#include<conio.h>

class demo {

int a[10],temp;

int i, min\_index,j,n;

public:

void get(); void asc\_sort();

void dsc\_sort(); void disp();

};

void demo::get() {

cout<<"\nEnter the array size:";

cin>>n;

cout<<"\nEnter the array element:";

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

cin>>a[i];

}

void demo::asc\_sort() {

for(i=1;i<=n-1;i++) {

min\_index=i;

for(j=i+1;j<=n;j++) {

if(a[j]<a[min\_index])

min\_index=j;

}

if(min\_index!=i) {

temp=a[min\_index];

a[min\_index]=a[i];

a[i]=temp;

}

}

}

void demo::dsc\_sort() {

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

min\_index=i;

for(j=i+1;j<=n;j++) {

if(a[j]>a[min\_index])

min\_index=j;

}

if(min\_index!=i)

{

temp=a[min\_index];

a[min\_index]=a[i];

a[i]=temp;

}

}

}

void demo::disp()

{

cout<<"\n The array element are";

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

cout<<a[i]<<"\t";

}

void main()

{

clrscr();

demo d;

d.get();

d.disp();

d.asc\_sort();

cout<<"\nAfter ascending sort:";

d.disp();

d.dsc\_sort();

cout<<"\n After Descending sort:";

d.disp();

getch();

}

**#STACK\_USING\_ARRAY.CPP**

#include<iostream.h>

#include<conio.h>

#include<process.h>

int n;

class stack

{

private:

int s[10],top,ele,i;

public:

stack()

{

top=-1;

}

void push();

void dis();

void pop();

void peep();

void change();

};

void stack::push()

{

if(top>=n-1)

cout<<"\nStack is overflow:";

else

{

cout<<"\nEnter element:";

cin>>ele;

top++;

s[top]=ele;

}

}

void stack::dis()

{

if(top==-1)

{

cout<<"\n Stack is Empty";

}

else

{

cout<<"\nElements in stack are:\n";

for(i=top;i>=0;i--)

cout<<s[i]<<"\t";

}

}

void stack::pop()

{

if(top==-1)

{

cout<<"\n Underflow";

}

else

{

cout<<"\n pop ele is:"<<s[top];

top--;

}

}

void stack::peep()

{

cout<<"\nEnter position:";

cin>>i;

if((top-i+1)<0)

{

cout<<"\nUnderflow";

}

else

{

cout<<"\nPeep ele is "<<s[top-i+1];

}

}

void stack::change()

{

cout<<"\nEnter position "; cin>>i;

if((top-i+1)<0)

{

cout<<"\nUnderFlow:";

}

else

{

cout<<"\nPeep ele is:"<<s[top-i+1];

}

}

void main()

{

clrscr();

stack s;

cout<<"Enter size of stack";

cin>>n;

int ch;

cout<<"\n1. Push 2.Display 3.Pop 4.Peep 5.Change 6.Exit\n";

while(ch!=6)

{

cout<<"\nEnter ch :";

cin>>ch;

switch(ch)

{

case 1: s.push(); break;

case 2: s.dis(); break;

case 3: s.pop();break;

case 4: s.peep(); break;

case 5: s.change(); break;

case 6: exit(0);

}

}

getch();

}

**#STACK\_USING\_LINKED.CPP**

#include<conio.h> #include<iostream.h>

#include<process.h>

class stack {

int info, ele;

stack \*node,\*link,\*top;

public:

stack() {

top=NULL;

}

void insert(); void del();

void dis();

};

void stack::insert() {

node=new stack;

cout<<"\nEnter Info:";

cin>>ele;

node->info=ele;

node->link=NULL;

if(top==NULL) {

top=node;

}

Else {

node->link=top;

top=node;

} }

void stack::del() {

if(top==NULL) {

cout<<"\nUnderflow:";

}

else

{

cout<<"\nDeleted Element is:"<<top->info;

top=top->link;

}

}

void stack::dis()

{

stack \*move;

move=top;

while(move!=NULL)

{

cout<<"\t"<<move->info;

move=move->link;

}

}

void main()

{

clrscr();

int ch;

stack s;

cout<<"\n1.Insert 2.Show 3.Delete 4.Exit";

while(ch!=4)

{

cout<<"\nEnter Choice"; cin>>ch;

switch(ch)

{

case 1: s.insert(); break;

case 2: s.dis(); break;

case 3: s.del(); break;

case 4:exit(0);

}

}

getch();

}