**DISCRETE PRACTICAL**

QUES 1:

#include<iostream>

#include<math.h>

using namespace std;

class set

{

int \*arr;

int size;

public:

set();

void input();

void unique();

void print();

int cardinality();

void powerset();

bool check\_avl(int a);

};

set::set()

{

arr=new int[0];

size=0;

return;

}

void set::input()

{

int n;

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

cin>>n;

size=n;

arr=new int[size];

cout<<"Enter the values of the set:\n";

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

cin >>arr[i];

return;

}

void set::unique()

{

for(int i=0;i<size;++i)

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

{

if(arr[i]==arr[j])

{

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

arr[k]=arr[k+1];

--size;

}

else

++j;

}

}

void set::print()

{

cout<<"{";

for(int i=0;i<size;i++)

{

cout<<arr[i];

if((i+1)!=size)

cout<<",";

}

cout<<"}"<<endl;

}

int set::cardinality()

{

return size;

}

void set::powerset()

{

int count=pow(2,size);

int temp;

cout<<"{ {},";

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

{

temp=i;

int counter=-1;

cout<<"{";

for(int j=0;j<size;j++)

{

if(temp&1)

{

counter++;

}

temp=temp>>1;

}

temp=i;

for(int j=0;j<size;j++)

{

if(temp&1)

{

cout<<arr[j];

if(counter!=0)

{

cout<<",";

counter--;

}

}

temp=temp>>1;

}

cout<<"}";

if((i+1)!=count)

cout<<",";

}

cout<<"}";

}

bool set::check\_avl(int a)

{

for(int i=0;i<size;i++)

{

if(a==arr[i])

return true;

}

return false;

}

int main()

{

set s;

s.input();

cout<<"The entered set is:\n";

s.unique();

s.print();

int choice;

for(char ch='y';ch=='y'||ch=='Y';)

{

cout<<"choose the task to perform: \n1.Find the cardinality of the set\n2.Check "

<<"whether input number belongs to the set or not\n3.Find the power set of the set.\n";

cin>>choice;

switch(choice)

{

case 1:

cout<<"The cardinality of the set is: "<<s.cardinality()<<endl;

break;

case 2:

cout<<"Enter a number to check: ";

int n;

cin>>n;

if(s.check\_avl(n))

cout<<"The entered number exist in the given set."<<endl;

else

cout<<"The entered number does not exist in the given set."<<endl;

break;

case 3:

cout<<"The power set of the given set is:"<<endl;

s.powerset();

cout<<endl;

break;

default:

cout<<"Error100:you have chose a wrong choice."<<endl;

}

cout<<endl<<"Do you want to do some other operations!!!(Y/N)"<<endl;

cin>>ch;

}

cout<<endl<<"The program executed successfully\nTHANK YOU!!!!";

}

QUES 2:

#include<iostream>

#include<math.h>

using namespace std;

class set

{

int \*arr;

int size;

public:

set();

void input();

void unique();

void print();

int cardinality();

bool check\_avl(int a);

bool sub\_set(set s);

set set\_union(set s);

set set\_intersection(set s);

set set\_cmpl(set s);

set set\_difference(set s);

set sym\_Diff(set s);

void cartesian\_Product(set s);

};

set::set()

{

arr=new int[0];

size=0;

return;

}

void set::input()

{

int n;

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

cin>>n;

size=n;

arr=new int[size];

cout<<"Enter the values of the set:\n";

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

cin >>arr[i];

return;

}

void set::unique()

{

for(int i=0;i<size;++i)

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

{

if(arr[i]==arr[j])

{

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

arr[k]=arr[k+1];

--size;

}

else

++j;

}

}

void set::print()

{

cout<<"{";

for(int i=0;i<size;i++)

{

cout<<arr[i];

if((i+1)!=size)

cout<<",";

}

cout<<"}"<<endl;

}

int set::cardinality()

{

return size;

}

bool set::check\_avl(int a)

{

for(int i=0;i<size;i++)

{

if(a==arr[i])

return true;

}

return false;

}

bool set::sub\_set(set s)

{

set temp=\*this;

for(int i=0;i<s.size;i++)

{

if(temp.check\_avl(s.arr[i]))

{

}

else

return false;

}

return true;

}

set set::set\_union(set s)

{

set temp;

temp.size=size+s.size;

temp.arr=new int[temp.size];

for(int i=0;i<size;i++)

{

temp.arr[i]=arr[i];

}

for(int j=0;j<s.size;j++)

{

temp.arr[size+j]=s.arr[j];

}

temp.unique();

return temp;

}

set set::set\_intersection(set s)

{

set temp,s1,s2;

int n,j=0;

if(size>s.size)

{

n=s.size;

s1=\*this;

s2=s;

}

else

{

n=size;

s1=s;

s2=\*this;

}

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

{

if(s1.check\_avl(s2.arr[i]))

{

temp.size++;

}

}

temp.arr=new int[temp.size];

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

{

if(s1.check\_avl(s2.arr[i]))

{

temp.arr[j++]=s2.arr[i];

}

}

return temp;

}

set set::set\_cmpl(set s)

{

set temp,s1,s2;

int n,j=0;

if(size>s.size)

{

n=size;

s1=\*this;

s2=s;

}

else

{

n=s.size;

s1=s;

s2=\*this;

}

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

{

if(s2.check\_avl(s1.arr[i]))

{

}

else

{

temp.size++;

}

}

temp.arr=new int[temp.size];

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

{

if(s2.check\_avl(s1.arr[i]))

{

}

else

{

temp.arr[j++]=s1.arr[i];

}

}

return temp;

}

set set :: set\_difference(set s){

int j,k=0,m=0;

set temp;

for(int i=0; i<size; i++){

for(j=0; j<s.size; j++)

if(arr[i]==s.arr[j])

break;

if(j==s.size)

k++;

}

temp.size=k;

temp.arr=new int[temp.size];

for(int i=0; i<size; i++){

for(j=0; j<s.size; j++)

if(arr[i]==s.arr[j])

break;

if(j==s.size)

temp.arr[m++]=arr[i];

}

return temp;

}

set set :: sym\_Diff(set s){

int i,j,k=0,m=0;

set temp;

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

for(j=0; j<s.size; j++)

if(arr[i]==s.arr[j])

break;

if(j==s.size)

k++;

}

for(int i=0; i<s.size; i++){

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

if(s.arr[i]==arr[j])

break;

if(j==size)

k++;

}

temp.size=k;

temp.arr=new int[temp.size];

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

for(j=0; j<s.size; j++)

if(arr[i]==s.arr[j])

break;

if(j==s.size)

temp.arr[m++]=arr[i];

}

for(int i=0; i<s.size; i++){

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

if(s.arr[i]==arr[j])

break;

if(j==size)

temp.arr[m++]=s.arr[i];

}

return temp;

}

void set :: cartesian\_Product(set s){

cout<<"{";

for(int i=0; i<size; i++)

for(int j=0; j<s.size; j++)

cout<<" ("<<arr[i]<<","<<s.arr[j]<<") ";

cout<<"}";

cout<<"\n";

}

int main()

{

set s1,s2,z,U;

cout<<"For the first set:"<<endl<<endl;

s1.input();

cout<<"The entered array is:\n";

s1.print();

cout<<"The entered set is:\n";

s1.unique();

s1.print();

cout<<"For the second set:"<<endl<<endl;

s2.input();

cout<<"The entered array is:\n";

s2.print();

cout<<"The entered set is:\n";

s2.unique();

s2.print();

int choice;

for(char ch='y';ch=='y'||ch=='Y';)

{

cout<<"choose the task to perform: \n1.Check whether the first set is subset of second set or not.\n2.Check "

<<"whether the second set is subset of first set or not.\n3.Find the union of the sets.\n4.Find the intersection "

<<"of the sets\n5.Complement of set\n6.Set difference of sets\n7.symmetric difference of sets\n8.Cartesian product of sets\n9.Exit\n";

cin>>choice;

switch(choice)

{

case 1:

if(s1.sub\_set(s2))

cout<<"The second set is the subset of the first set"<<endl;

else

cout<<"The second set is not the subset of the first set"<<endl;

break;

case 2:

if(s2.sub\_set(s1))

cout<<"The first set is the subset of the second set"<<endl;

else

cout<<"The first set is not the subset of the second set"<<endl;

break;

case 3:

z=s1.set\_union(s2);

cout<<"The union of first and second set is: "<<endl;

z.print();

break;

case 4:

z=s1.set\_intersection(s2);

cout<<"The intersection of first and second set is: "<<endl;

z.print();

break;

case 5:

cout<<"For the universal set:"<<endl<<endl;

U.input();

cout<<"The entered array is:\n";

U.print();

cout<<"The entered set is:\n";

U.unique();

U.print();

cout<<"Choose one from given options\n1.Complement of first set\n2.Compliment of second set\n";

cin>>choice;

switch(choice)

{

case 1:

if(U.sub\_set(s1))

{

z=s1.set\_cmpl(U);

z.cardinality();

cout<<"Complement of first set is:"<<endl;

z.print();

}

else

cout<<"the set is not the subset of universal set"<<endl;

break;

case 2:

if(U.sub\_set(s2))

{

z=s2.set\_cmpl(U);

z.cardinality();

cout<<"Complement of second set is:"<<endl;

z.print();

}

else

cout<<"the set is not the subset of universal set"<<endl;

break;

default:

cout<<"Error100:you have chose a wrong choice."<<endl;

}

break;

case 6:

cout<<"Choose one from given options\n1.Set difference of second set from first set\n2.Set difference of first set from second set\n";

cin>>choice;

switch(choice)

{

case 1:

z=s1.set\_difference(s2);

cout<<"The output set S1-S2=:"<<endl;

z.print();

break;

case 2:

z=s2.set\_difference(s1);

cout<<"The output set s2-s1=:"<<endl;

z.print();

break;

default:

cout<<"Error100:you have chose a wrong choice."<<endl;

}

break;

case 7 :

cout<<"Symmetric Difference of set A and Set B :\n";

z=s1.sym\_Diff(s2);

z.print();

break;

case 8 :

cout<<"Cartesian Product AXB :\n";

s1.cartesian\_Product(s2);

cout<<"Cartesian Product BXA :\n";

s2.cartesian\_Product(s1);

break;

case 9 :

cout<<endl<<"The program executed successfully\nTHANK YOU!!!!";

exit(0);

break;

default:

cout<<"Error100:you have chose a wrong choice."<<endl;

}

cout<<endl<<"Do you want to do some other operations!!!(Y/N)"<<endl;

cin>>ch;

}

cout<<endl<<"The program executed successfully\nTHANK YOU!!!!";

}

QUES 3:

#include<iostream>

using namespace std;

class relation

{

int\*\* ar;

int n;

public:

void input();

bool reflexive();

bool symmetric();

bool antisymmetric();

bool transitive();

void display();

};

void relation::input()

{

cout<<"Enter the size of set as an array : ";

cin>>n;

ar=new int\*[n];

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

ar[i]=new int[n];

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

{

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

{

ar[i][j]=0;

}

}

int m;

cout<<"Enter the no of relations you want:";

cin>>m;

int a[m],b[m];

cout<<"Enter the relations:";

for(int i=0;i<m;i++)

cin>>a[i]>>b[i];

for(int i=0;i<m;i++)

ar[(a[i]-1)][(b[i]-1)]=1;

}

void relation::display()

{

cout<<"\nThe Relation Matrix is:\n";

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

{

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

{

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

}

cout<<endl;

}

}

bool relation::reflexive()

{

int f=1;

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

{

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

{

if(i==j && ar[i][j]!=1)

{

f=0;

break;

}

}

if(f==0)

break;

}

if(f==1)

return true;

else

return false;

}

bool relation::symmetric()

{

int f=1;

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

{

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

{

if(ar[i][j]!=ar[j][i])

{

f=0;

break;

}

}

if(f==0)

break;

}

if(f==1)

return true;

else

return false;

}

bool relation::transitive()

{

int f=1;

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

{

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

{

if(ar[i][j]==1)

{

for(int x=0;x<n;x++)

{

if(ar[j][x]==1 && ar[i][x]!=1)

{

f=0;

break;

}

}

}

if(f==0)

break;

}

if(f==0)

break;

}

if(f==1)

return true;

else

return false;

}

bool relation::antisymmetric()

{

int f=1;

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

{

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

{

if(ar[i][j]==1 && ar[j][i]==1 && i!=j)

{

f=0;

break;

}

}

if(f==0)

break;

}

if(f==1)

return true;

else

return false;

}

int main()

{

relation r1;

r1.input();

r1.display();

if(r1.reflexive())

cout<<"\nRelation is Reflexive"<<endl;

if(r1.symmetric())

cout<<"\nRelation is Symmetric"<<endl;

if(r1.transitive())

cout<<"\nRelation is Transitive"<<endl;

if(r1.antisymmetric())

cout<<"\nRelation is Antisymmetric"<<endl;

return 0;

}

QUES 4:

#include<iostream>

using namespace std;

class relation

{

int\*\* ar;

int n;

public:

void input();

bool reflexive();

bool symmetric();

bool antisymmetric();

bool transitive();

void display();

};

void relation::input()

{

cout<<"enter the size of set as an array : ";

cin>>n;

ar=new int\*[n];

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

ar[i]=new int[n];

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

{

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

{

ar[i][j]=0;

}

}

int m;

cout<<"Enter the no of relations you want:";

cin>>m;

int a[m],b[m];

cout<<"Enter the relation:";

for(int i=0;i<m;i++)

cin>>a[i]>>b[i];

for(int i=0;i<m;i++)

ar[(a[i]-1)][(b[i]-1)]=1;

}

void relation::display()

{

cout<<"\nThe Relation Matrix is:\n";

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

{

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

{

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

}

cout<<endl;

}

}

bool relation::reflexive()

{

int f=1;

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

{

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

{

if(i==j && ar[i][j]!=1)

{

f=0;

break;

}

}

if(f==0)

break;

}

if(f==1)

return true;

else

return false;

}

bool relation::symmetric()

{

int f=1;

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

{

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

{

if(ar[i][j]!=ar[j][i])

{

f=0;

break;

}

}

if(f==0)

break;

}

if(f==1)

return true;

else

return false;

}

bool relation::transitive()

{

int f=1;

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

{

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

{

if(ar[i][j]==1)

{

for(int x=0;x<n;x++)

{

if(ar[j][x]==1 && ar[i][x]!=1)

{

f=0;

break;

}

}

}

if(f==0)

break;

}

if(f==0)

break;

}

if(f==1)

return true;

else

return false;

}

bool relation::antisymmetric()

{

int f=1;

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

{

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

{

if(ar[i][j]==1 && ar[j][i]==1 && i!=j)

{

f=0;

break;

}

}

if(f==0)

break;

}

if(f==1)

return true;

else

return false;

}

int main()

{

relation r;

r.input();

r.display();

if(r.reflexive()&&r.symmetric()&&r.transitive())

cout<<"The given relation is Equivalence";

else if(r.reflexive()&&r.antisymmetric()&&r.transitive())

cout<<"The given relation is a partial order relation";

else

cout<<"The relation is neither equivalence nor partial order relation";

return 0;

}

QUES 5:

#include<iostream>

using namespace std;

int fib(int x)

{

if((x==1)||(x==0))

{

return x;

}

else

{

return(fib(x-1)+fib(x-2));

}

}

int main()

{

int x, i=0;

cout<<"Enter the number of terms of the series: ";

cin>>x;

cout<<"\nFibonaci series upto "<<x<<"terms is: ";

while(i<x)

{

cout<<" "<<fib(i);

i++;

}

return 0;

}

QUES 6:

#include<iostream>

using namespace std;

int hanoitower(int n)

{

int a=1;

if(n>1)

a = 2\*hanoitower(n-1)+1;

else

return a;

}

int main()

{

int n;

cout<<"Enter the number of stacks on Tower of Hanoi: "<<endl;

cin>>n;

cout<<"The minimum number of shifts of stack is: "<<hanoitower(n);

return 0;

}

QUES 7:

#include<iostream>

using namespace std;

void input(int arr[],int n)

{

cout<<"Enter the values of the array: "<<endl;

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

cin>>arr[i];

return;

}

void display(int arr[],int n)

{

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

cout<<arr[i]<<" ";

cout<<endl;

return;

}

int binarysearch(int a[],int start,int end,int element)

{

if(start>end)

return -1;

int mid=(start+end)/2;

if(a[mid]==element)

return mid;

else if(element<a[mid])

binarysearch(a,start,mid-1,element);

else

binarysearch(a,mid+1,end,element);

}

void insertionsort(int arr[],int n)

{

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

{

int key=arr[i];

int j=i-1;

while(j>=0&&arr[j]>key)

{

arr[j+1]=arr[j];

j--;

}

arr[j+1]=key;

}

return;

}

int main()

{

int n;

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

cin>>n;

int arr[n];

input(arr,n);

cout<<"The entered array is: "<<endl;

display(arr,n);

insertionsort(arr,n);

cout<<"\nInput array after sorting: \n\n";

display(arr,n);

cout<<"\nEnter the element to be searched: ";

int k;

cin>>k;

cout<<k<<" is found at Index "<<(binarysearch(arr,0,n-1,k)+1);

return 0;

}

QUES 8:

#include<iostream>

using namespace std;

void input(int arr[],int n)

{

cout<<"Enter the values of the array: "<<endl;

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

cin>>arr[i];

return;

}

void display(int arr[],int n)

{

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

cout<<arr[i]<<" ";

cout<<endl;

return;

}

void swap(int \*a,int \*b)

{

int temp =\*b;

\*b=\*a;

\*a=temp;

return;

}

void bubblesort(int arr[],int n)

{

cout<<"The sorting process start: \n";

for(int i=0;i<n-1;i++)

{

bool flag=true;

for(int j=0;j<n-i-1;j++)

{

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

{

swap(&arr[j],&arr[j+1]);

flag=false;

}

display(arr,n);

}

if(flag)

{

break;

}

}

cout<<"The sorting process ended.....\n";

return;

}

int main()

{

int n;

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

cin>>n;

int arr[n];

input(arr,n);

cout<<"The entered array is: "<<endl;

display(arr,n);

bubblesort(arr,n);

cout<<"Array after bubble sorting: "<<endl;

display(arr,n);

return 0;

}

QUES 9:

#include<iostream>

using namespace std;

void input(int arr[],int n)

{

cout<<"Enter the values of the array: "<<endl;

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

cin>>arr[i];

return;

}

void display(int arr[],int n)

{

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

cout<<arr[i]<<" ";

cout<<endl;

return;

}

void insertionsort(int arr[],int n)

{

cout<<"The sorting process started: \n";

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

{

int key=arr[i];

int j=i-1;

while(j>=0&&arr[j]>key)

{

arr[j+1]=arr[j];

j--;

}

arr[j+1]=key;

display(arr,n);

}

cout<<"The sorting process ended....\n";

return;

}

int main()

{

int n;

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

cin>>n;

int arr[n];

input(arr,n);

cout<<"The entered array is: "<<endl;

display(arr,n);

insertionsort(arr,n);

cout<<"\nInput array after insertion sorting: \n\n";

display(arr,n);

return 0;

}

QUES 10:

#include<iostream>

using namespace std;

void permutation\_without\_repetition(int A[],int size,int b)

{

if(b==size-1)

{

for(int i=0;i<size;i++)

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

cout<<endl;

}

else

{

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

{

swap(A[b],A[i]);

permutation\_without\_repetition(A,size,b+1);

swap(A[b],A[i]);

}

}

}

void permutation\_with\_repetition(int A[],int B[],int size,int x)

{

if(x==size)

{

for(int i=0;i<size;i++)

cout<<B[i]<<" ";

cout<<endl;

}

else

{

for(int i=0;i<size;i++)

{

B[x]=A[i];

permutation\_with\_repetition(A,B,size,x+1);

}

}

}

void swap(int \*x,int \*y)

{

int temp;

temp = \*x;

\*x = \*y;

\*y = temp;

}

void display(int A[],int size){

cout<<"\n The given set is:{";

for(int i=0;i<size;i++)

{

if(i==0)

cout<<A[i];

else

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

}

cout<<"}";

}

int main()

{

char choice='y';

int n,b=0,ch;

int\*A,\*B;

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

cin>>n;

A=new int [n];

B=new int [n];

cout<<"\nEnter the elements of set: \t";

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

cin>>A[i];

display(A,n);

do

{

cout<<"\n PERMUTATION MENU:";

cout<<"\n1.Permutation without repetition";

cout<<"\n2.Permutation with repetition";

cout<<"\n Enter your choice\t: ";

cin>>ch;

switch(ch)

{

case 1:

{

permutation\_without\_repetition(A,n,b);

break;

}

case 2:

{

permutation\_with\_repetition(A,B,n,b);

break;

}

default :cout<<"\n WRONG INPUT!!!";

} cout<<"\n Do you want to continue <enter y or Y to continue> ";

cin>>choice;

}while(choice=='y'||choice=='Y');

return 0;

}

QUES 11:

#include<iostream>

using namespace std;

int nprfun(int n,int r)

{

if(r>n)return 0;

else if (r==0) return 1;

else return n\*nprfun(n-1,r-1);

}

int ncrfun(int n,int r)

{

if(r>n) return 0;

else if(n==0||r==0||n==r) return 1;

else return ncrfun(n-1,r-1)+ncrfun(n-1,r);

}

int main()

{

int n,r;

cout<<"Enter the value of n: ";

cin>>n;

cout<<"Enter the value of r: ";

cin>>r;

cout<<"The permutation of given n and r is: "<<nprfun(n,r);

cout<<endl<<"The combination of given n and r is: "<<ncrfun(n,r);

return 0;

}

QUES 12:

#include<iostream>

using namespace std;

void display(int b[],int n)

{

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

cout<<b[i]<<" ";

}

int combos(int b[],int k,int n,int s)

{

if(k==0)

{

b[k]=s;

display(b,n);

cout<<"\n";

return 0;

}

for(int i=0;i<=s;i++)

{

b[k]=i;

combos(b,k-1,n,s-i);

}

}

int main()

{

int s,n;

cout<<"Enter the number of groups: ";

cin>>n;

cout<<"Enter the sum to be obtained: ";

cin>>s;

int b[n];

combos(b,n-1,n,s);

return 0;

}

QUES 13:

#include<iostream>

using namespace std;

int main()

{

int x[4];

int y[4];

int disjunction[4];

int conjuction[4];

int NOR[4];

int NAND[4];

int Cond[4];

int Bicond[4];

int Negative\_x[4];

int Negative\_y[4];

char ch;

do

{

for(int i=0;i<4;i++)

{

cout<<"Enter the"<<" "<<(i+1)<<" "<<"value of x and y"<<endl;

cin>>x[i]>>y[i];

disjunction[i] = x[i]|y[i];

conjuction[i] = x[i]&y[i];

NOR[i] = !disjunction[i];

NAND[i] = !conjuction[i];

Cond[i] = !x[i]|y[i];

Bicond[i] = ((!x[i]|y[i])&(!y[i]|x[i]));

Negative\_x[i] = !x[i];

Negative\_y[i] = !y[i];

}

cout<<"x |y |or |and|NOR|NAND|Cond|Bicond|Negx|Negy"<<endl;

for(int i=0;i<4;i++)

{

cout<<x[i]<<" | "<<y[i]<<" | "<<disjunction[i]<<" | "<<conjuction[i]<<" | "<<NOR[i]<<" | "<<NAND[i]<<" | ";

cout<<Cond[i]<<" |"<<Bicond[i]<<" |"<<Negative\_x[i]<<" |"<<Negative\_y[i]<<endl;

}

cout<<"do you want to continue(y/n)\n";

cin>>ch;

}while(ch=='y');

return 0;

}

QUES 14:

#include<iostream>

using namespace std;

int T1(int n)

{

while(n>=0)

{

if(n==0)

return 1;

else

return (T1(n-1)+n);

}

}

int T2(int n)

{

while(n>=0)

{

if(n==0)

return 1;

else

return (T2(n-1)+(n^2));

}

}

int T3(int n)

{

while(n>=1)

{

if(n==1)

return 1;

else

return (2\*T3(n/2)+n);

}

}

//Staring of main

int main()

{

int n;

cout<<"\n Enter n : ";

cin>>n;

cout<<"\n Following are the recurrence relation ..... ";

cout<<"\n 1. T1(x-1)+x";

cout<<"\n 2. T2(x-1)+(x^2)";

cout<<"\n 3. 2\*T3(x/2)+x";

cout<<"\n Enter your choice :";

int r;

int ch;

cin>>ch;

switch(ch)

{

case 1:

{

r=T1(n);

cout<<"\n The answer is : "<<r;

break;

}

case 2:

{

r=T2(n);

cout<<"\n The answer is : "<<r;

break;

}

case 3:

{

r=T3(n);

cout<<"\n The answer is : "<<r;

break;

}

}

}

QUES 15:

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

int degree,x,sum=0;

char ch;

do

{

cout<<"Enter the degree of the polynomial :";

cin>>degree;

int arr[degree];

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

{

cout<<"Enter the coefficient of degree "<<i<<" :";

cin>>arr[i];

}

cout<<"The required polynomial is :";

cout<<arr[degree]<<"x^"<<degree;

for(int i=degree-1;i>0;i--)

{

if(arr[i]>0)

cout<<"+"<<arr[i]<<"x^"<<i;

else

cout<<"-"<<arr[i]<<"x^"<<i;

}

cout<<"+"<<arr[0]<<"x^"<<0;

cout<<"\nEnter the value of x : ";

cin>>x;

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

{

sum+=(arr[i]\*pow(x,i));

}

cout<<"\nThe solution of this polynomial is :"<<sum;

char ch;

cout<<"\nDo you want to continue?(y/n):";

cin>>ch;

}while(ch=='y'||ch=='Y');

}

QUES 16:

#include<iostream>

using namespace std;

class relation

{

int\*\* ar;

int n;

public:

void input();

bool complete();

void display();

};

void relation::input()

{

cout<<"Enter the number of vertices in the graph : ";

cin>>n;

ar=new int\*[n];

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

ar[i]=new int[n];

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

{

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

{

ar[i][j]=0;

}

}

int m;

cout<<"Enter the no of edges :";

cin>>m;

int a[m],b[m];

cout<<"Enter the vertices having edges in between :";

for(int i=0;i<m;i++)

cin>>a[i]>>b[i];

for(int i=0;i<m;i++)

{

ar[(a[i]-1)][(b[i]-1)]=1;

ar[(b[i]-1)][(a[i]-1)]=1;

}

}

bool relation::complete()

{

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

{

if(ar[i][i]==1)

return false;

}

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

{

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

{

if(j!=i)

{

if(ar[i][j]==0)

return false;

}

}

}

return true;

}

void relation::display()

{

cout<<"\nThe Matrix for the graph:\n";

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

{

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

{

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

}

cout<<endl;

}

}

int main()

{

relation r;

r.input();

r.display();

if(r.complete())

cout<<"The graph is a complete graph....";

else

cout<<"The graph is not a complete graph...";

}

QUES 17:

#include<iostream>

using namespace std;

class relation

{

int\*\* ar;

int n;

public:

void input();

void indeg();

void outdeg();

void display();

};

void relation::input()

{

cout<<"Enter the number of vertices in the graph : ";

cin>>n;

ar=new int\*[n];

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

ar[i]=new int[n];

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

{

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

{

ar[i][j]=0;

}

}

int m;

cout<<"Enter number of edges:";

cin>>m;

int a[m],b[m];

cout<<"Enter vertices having edges in between : ";

for(int i=0;i<m;i++)

cin>>a[i]>>b[i];

for(int i=0;i<m;i++)

ar[(a[i]-1)][(b[i]-1)]=1;

}

void relation::indeg()

{

int arr[n];

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

arr[i]=0;

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

{

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

{

if(ar[i][j]==1)

arr[j]++;

}

}

cout<<"IN-degree of elements for the given graph is: \n";

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

cout<<"For "<<i+1<<" : "<<arr[i]<<endl;

return;

}

void relation::outdeg()

{

int arr[n];

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

arr[i]=0;

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

{

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

{

if(ar[i][j]==1)

arr[i]++;

}

}

cout<<"OUT-degree of elements for the given graph is: \n";

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

cout<<"For "<<i+1<<" : "<<arr[i]<<endl;

return;

}

void relation::display()

{

cout<<"\nThe Matrix for the graph is:\n";

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

{

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

{

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

}

cout<<endl;

}

}

int main()

{

relation r;

r.input();

r.display();

r.indeg();

r.outdeg();

}

QUES 18:

#include<iostream>

using namespace std;

class path

{

private:

int \*a,\*b;

int size,size\_v;

int \*\*matrix;

int source,destination;

void adjacency\_matrix(int x,int y)

{

matrix[x-1][y-1]=1;

}

int \*\*mat;

int \*\*mul;

int multiply()

{

for(int i=0;i<size\_v;i++)

for(int j=0;j<size\_v;j++)

{

mat[i][j]=0;

for(int k=0;k<size\_v;k++)

mat[i][j]+=matrix[i][k] \* mul[k][j];

}

for(int i=0;i<size\_v;i++)

for(int j=0;j<size\_v;j++)

mul[i][j]=mat[i][j];

}

public:

path()

{

a=NULL;

b=NULL;

size=0;

size\_v=0;

}

void initialize(int x)

{

size=x;

a = new int[size];

b = new int[size];

matrix = new int\*[size\_v];

for(int i = 0;i<size\_v;i++)

matrix[i] = new int[size\_v];

for(int i=0;i<size\_v;i++)

{

for(int j=0;j<size\_v;j++)

matrix[i][j]=0;

}

mat = new int\*[size\_v];

for(int i = 0;i<size\_v;i++)

mat[i] = new int[size\_v];

mul = new int\*[size\_v];

for(int i = 0;i<size\_v;i++)

mul[i] = new int[size\_v];

for(int i=0;i<size\_v;i++)

{

for(int j=0;j<size\_v;j++)

mul[i][j]=1;

}

}

void vertices()

{

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

cin>>size\_v;

cout<<"\n The vertex numbers are : { ";

for(int i=0;i<size\_v;i++)

{

cout<<i+1<<",";

}

cout<<" } \n";

}

void edges()

{

char ch;

int ct = -1;

cout<<"\n Enter the edges from the above mentioned vertex number . ";

for(int i=0;i<size;i++)

{

ct++;

do

{

cout<<"\n Enter the vertex of "<<ct+1<<" edge : ";

cin>>a[ct];

if(a[ct]>size\_v)

{

cout<<"\n You have entered wrong edge.PLEASE!! enter it from above mentioned vertex number.";

}

}while(a[ct]>size\_v);

do

{

cout<<"\n Enter the vertex of "<<ct+1<<"edge : ";

cin>>b[ct];

if(b[ct]>size\_v)

{

cout<<"\n You have entered wrong edge.PLEASE!! enter it from above mentioned vertex number.";

}

}while(b[ct]>size\_v);

adjacency\_matrix(a[ct],b[ct]);

}

cout<<"\n The edges are : { ";

for(int i=0;i<size;i++)

{

cout<<"("<<a[i]<<","<<b[i]<<")";

}

cout<<"}";

}

void p\_adjacency()

{

cout<<"\n The adjacency matrix is : \n ";

for(int i=0;i<size\_v;i++)

{

for(int j=0;j<size\_v;j++)

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

cout<<"\n";

}

}

void c\_path(int k)

{

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

cin>>source;

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

cin>>destination;

for(int i=0;i<k;i++)

multiply();

cout<<"\n Number of paths of length "<<k<<" : ";

cout<<mat[source-1][destination-1];

}

};

int main()

{

path p;

p.vertices();

int y;

cout<<"\n Enter the number of edges you want : ";

cin>>y;

p.initialize(y);

p.edges();

p.p\_adjacency();

int length;

cout<<"\n Enter the length of path : ";

cin>>length;

p.c\_path(length);

}

QUES 19:

#include<iostream>

using namespace std;

class relation

{

int\*\* ar;

int n;

public:

void input();

void deg();

void display();

};

void relation::input()

{

cout<<"Enter the number of vertices in the graph : ";

cin>>n;

ar=new int\*[n];

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

ar[i]=new int[n];

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

{

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

{

ar[i][j]=0;

}

}

int m;

cout<<"Enter the number of edges :";

cin>>m;

int a[m],b[m];

cout<<"Enter the vertices having edges in between :";

for(int i=0;i<m;i++)

cin>>a[i]>>b[i];

for(int i=0;i<m;i++)

ar[(a[i]-1)][(b[i]-1)]=1;

}

void relation::deg()

{

int arr[n];

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

arr[i]=0;

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

{

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

{

if(ar[i][j]==1)

arr[j]++;

}

}

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

arr[i]=0;

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

{

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

{

if(ar[i][j]==1)

arr[i]--;

}

}

cout<<"Degree of elements for the given graph is: \n";

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

{

if(arr[i]<0)

arr[i]\*=-1;

}

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

cout<<"For "<<i+1<<" : "<<arr[i]<<endl;

int c=0;

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

{

if((arr[i]%2)!=0)

c++;

}

if(c==0)

cout<<"There is Euler circuit exist\n";

else

cout<<"There is no Euler circuit exist\n";

if(c==2)

cout<<"There is a Euler path\n";

else

cout<<"There is no Euler path\n";

return;

}

void relation::display()

{

cout<<"\nThe Matrix of the graph is:\n";

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

{

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

{

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

}

cout<<endl;

}

}

int main()

{

relation r;

r.input();

r.display();

r.deg();

}

QUES 20:

#include<iostream>

using namespace std;

int main()

{

int m,l,k;

cout<<"Enter the degree of tree: \n";

cin>>m;

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

cin>>k;

l=k\*(m-1)+1;

cout<<"The number of leaf nodes is: "<<l<<endl;

}

….THE END….