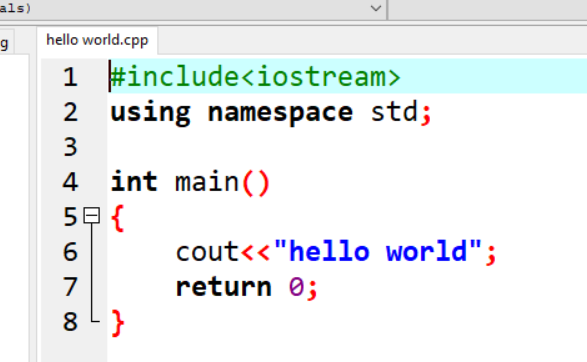
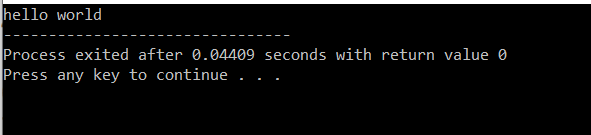
**EXPERIMENT 1.1**

**AIM-**print hello world

**PROGRAM-**



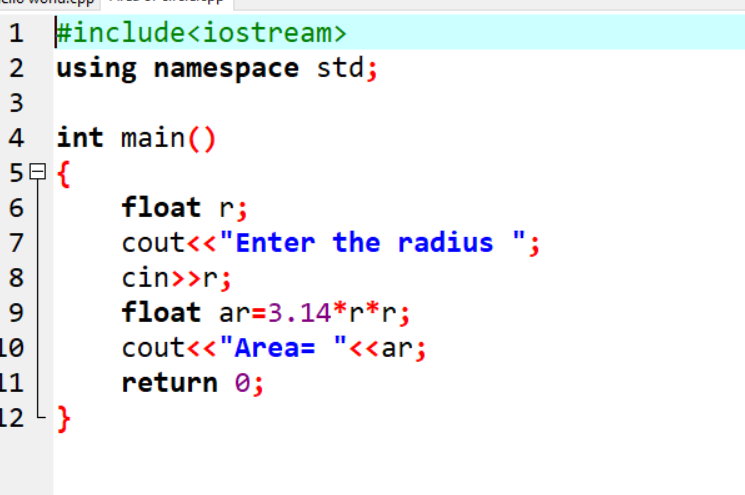
**OUTPUT-**



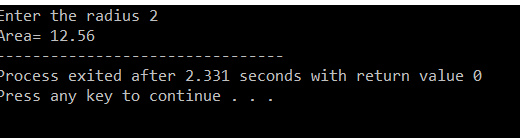
**EXPERIMENT 1.2**

**AIM-Find area of circle**

**PROGRAM-**



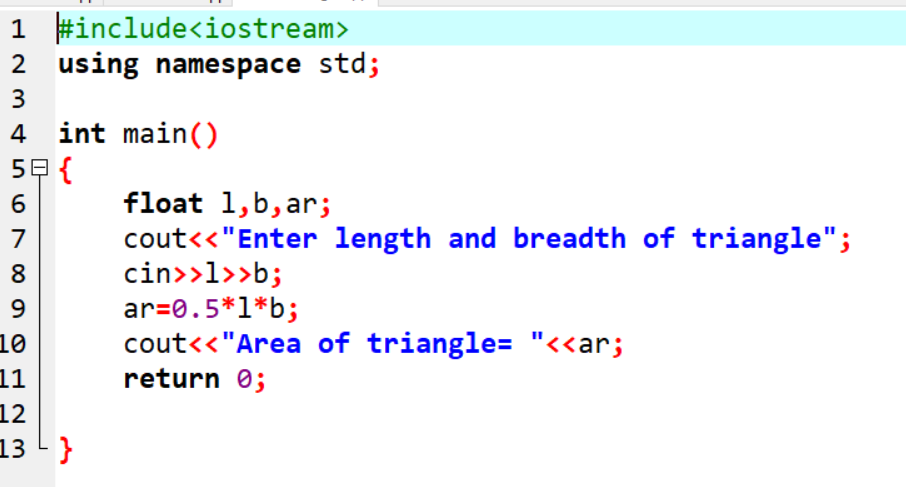
**OUTPUT-**



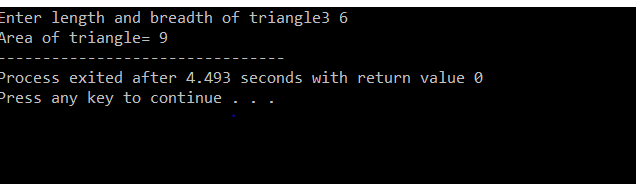
**EXPERIMENT 1.3**

**AIM-find the area of triangle**

**PROGRAM-**



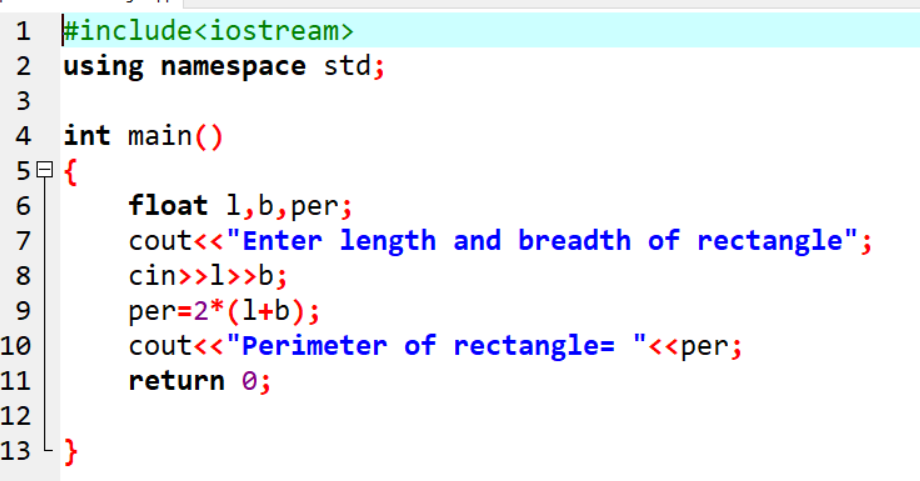
**OUTPUT-**



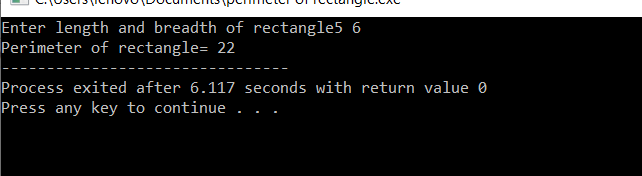
**EXPERIMENT 1.4**

**AIM-find perimeter of rectangle**

**PROGRAM-**



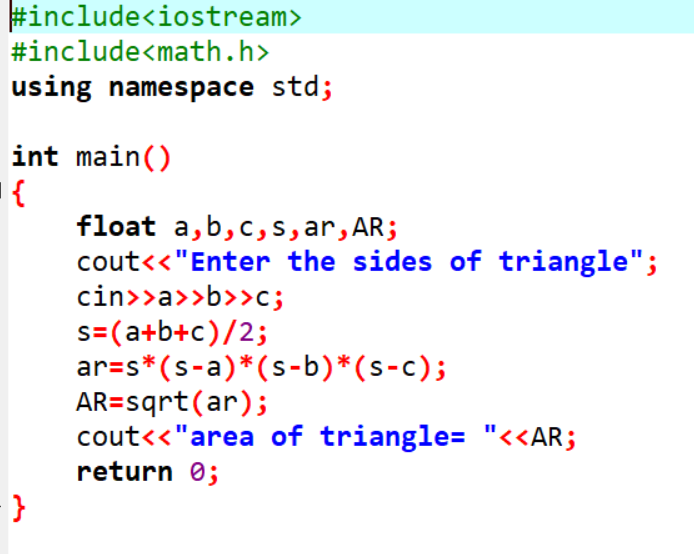
**OUTPUT-**



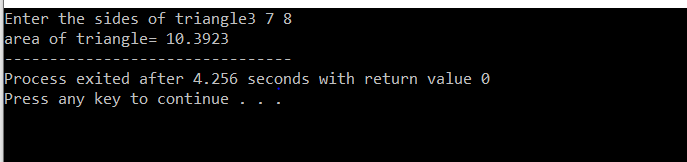
**EXPERIMENT 1.5**

**AIM-find area of triangle if three sides are given**

**PROGRAM-**



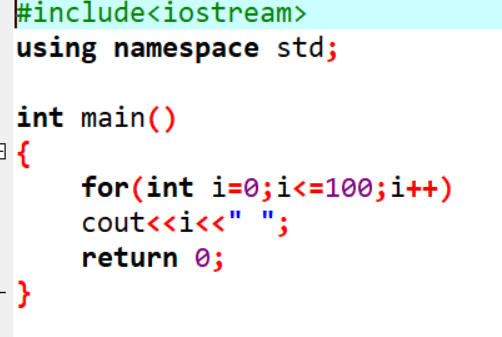
**OUTPUT-**



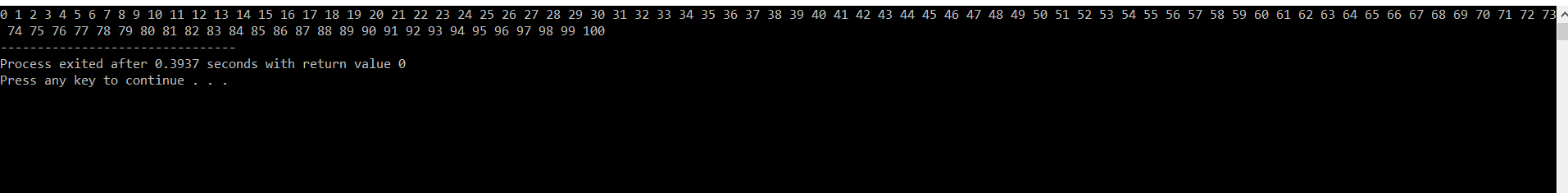
**EXPERIMENT 1.6**

**AIM-print natural numbers from 1 to 100**

**PROGRAM-**



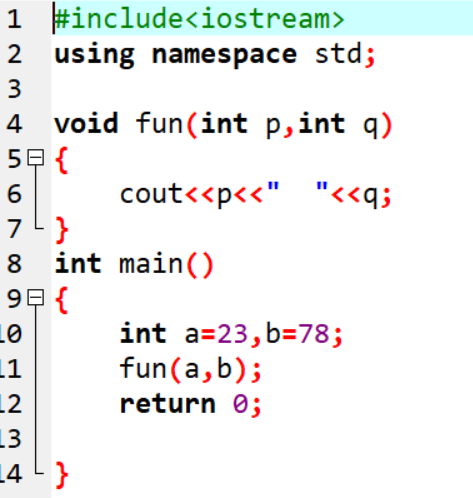
**OUTPUT-**



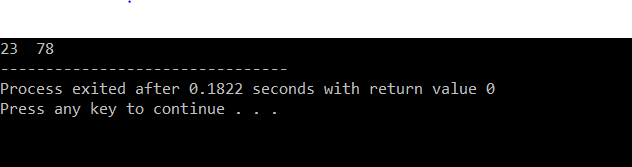
**EXPERIMENT 2.1**

**AIM-Call by value**

**PROGRAM-**



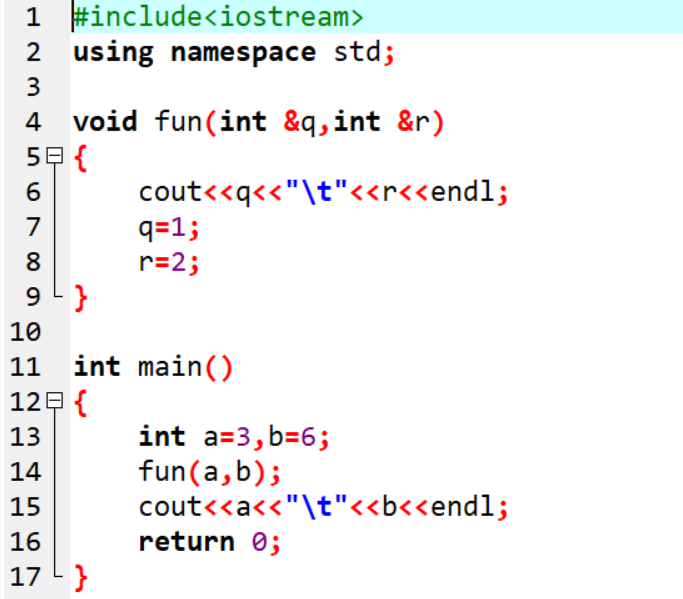
**OUTPUT-**



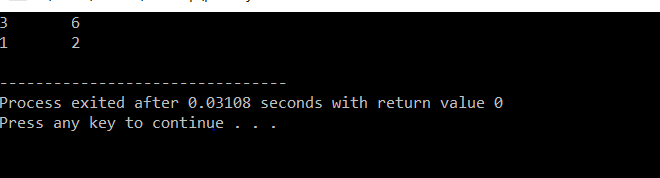
**EXPERIMENT 2.2**

**AIM-Call by reference**

**PROGRAM-**



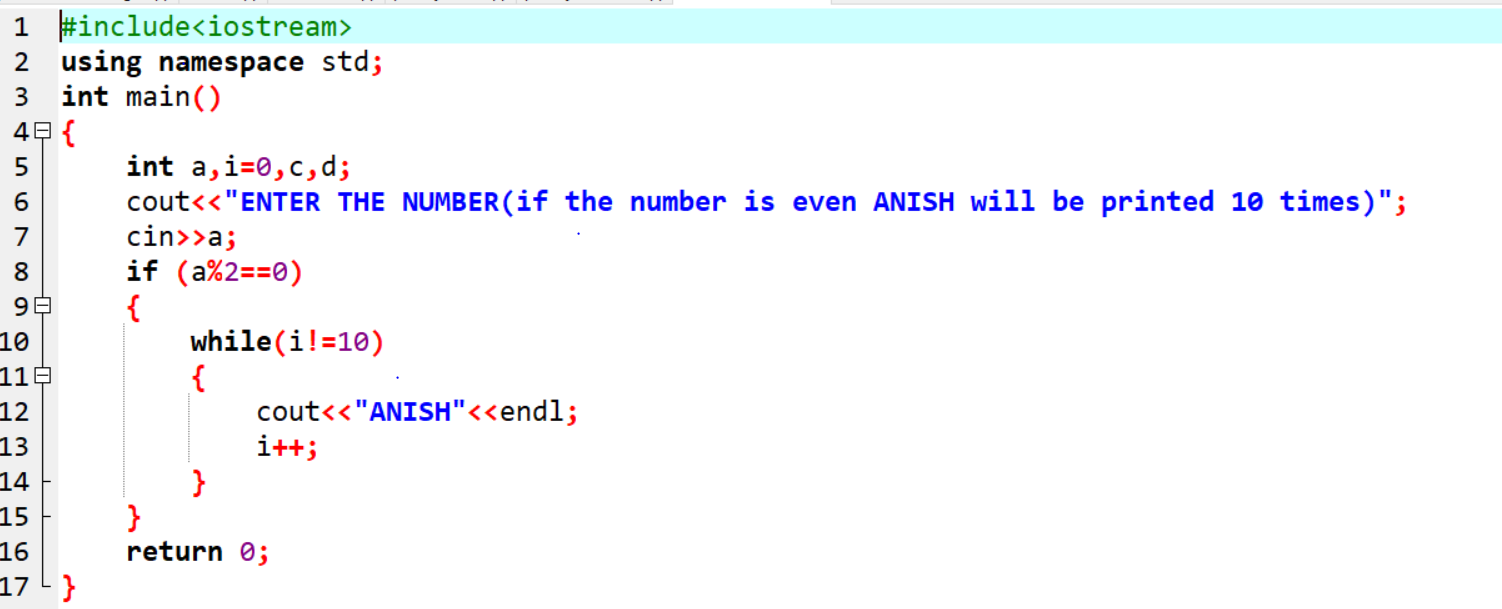
**OUTPUT-**



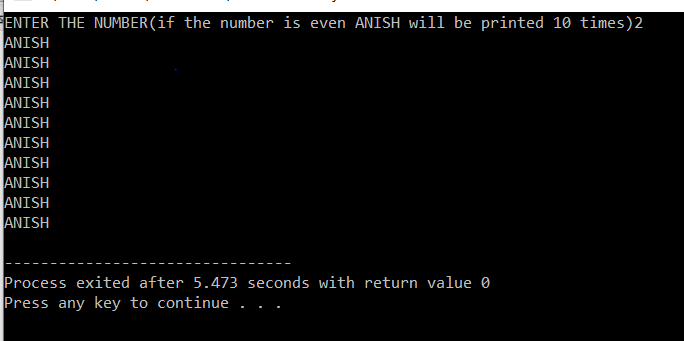
**EXPERIMENT 3.1**

**AIM-implement if loop**

**PROGRAM-**



**OUTPUT-**



**EXPERIMENT 3.2**

**AIM-impliment if else ladder**

**PROGRAM-**

**#include<iostream>**

**using namespace std;**

**int main()**

**{**

**float a;**

**cout<<"ENTER THE MARKS ";**

**cin>>a;**

**if(a<35)**

**{**

**cout<<"GRADE=F";**

**}**

**else if(a>=35&&a<40)**

**cout<<"GRADE=E"<<endl;**

**else if(a>=40&&a<45)**

**cout<<"GRADE=E+"<<endl;**

**else if(a>=45&&a<50)**

**cout<<"GRADE=D"<<endl;**

**else if(a>=50&&a<55)**

**cout<<"GRADE=D+"<<endl;**

**else if(a>=55&&a<60)**

**cout<<"GRADE=C"<<endl;**

**else if(a>=60&&a<65)**

**cout<<"GRADE=C+"<<endl;**

**else if(a>=65&&a<70)**

**cout<<"GRADE=B"<<endl;**

**else if(a>=70&&a<75)**

**cout<<"GRADE=B+"<<endl;**

**else if(a>=75&&a<80)**

**cout<<"GRADE=A"<<endl;**

**else if(a>=80&&a<90)**

**cout<<"GRADE=A+"<<endl;**

**else if(a>=90&&a<=100)**

**cout<<"GRADE=O"<<endl;**

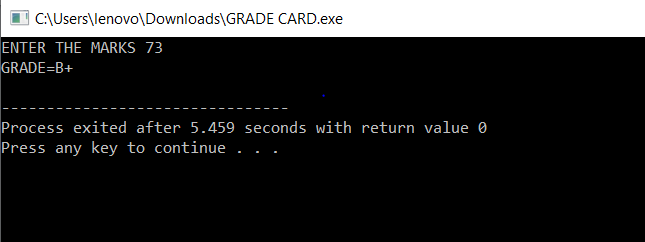
**else**

**cout<<"NOT VALID"<<endl;**

**return 0;**

**}**

**OUTPUT-**



**EXPERIMENT 3.3**

**AIM-implement switch case**

**PROGRAM-**

**#include<iostream>**

**using namespace std;**

**int main()**

**{**

**int a,b,c,d;**

**cout<<"Enter the Day number ";**

**cin>>a;**

**switch(a)**

**{**

**case 1: cout<<"MONDAY"<<endl;**

**return 0;**

**case 2: cout<<"TUESDAY"<<endl;**

**return 0;**

**case 3: cout<<"WEDNESDAY"<<endl;**

**return 0;**

**case 4: cout<<"THURSDAY"<<endl;**

**return 0;**

**case 5: cout<<"FRIDAY"<<endl;**

**return 0;**

**case 6: cout<<"SATURDAY"<<endl;**

**return 0;**

**case 7: cout<<"SUNDAY"<<endl;**

**return 0;**

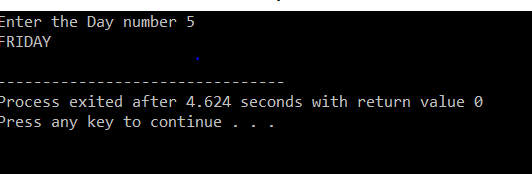
**default: cout<<"NOT VALID"<<endl;**

**}**

**return 0;**

**}**

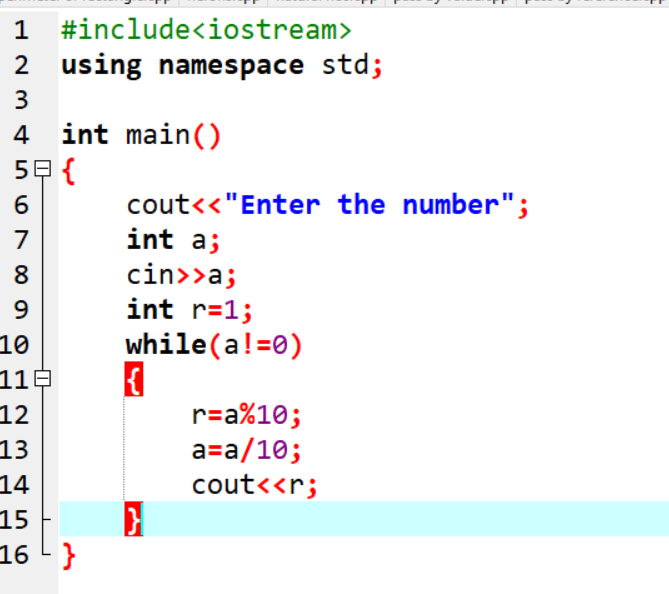
**OUTPUT-**



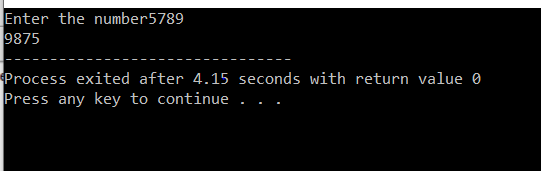
**EXPERIMENT 3.4**

**AIM-implement while loop**

**PROFRAM-**



**OUTPUT-**



**EXPERIMENT 3.5**

**AIM-implement do while loop**

**PROGRAM-**

**#include<iostream>**

**using namespace std;**

**class inchToFeet{**

**int input;**

**int inch;**

**int feet;**

**public:**

**void setinput(int input)**

**{**

**this->input=input;**

**}**

**inchToFeet(int input)**

**{**

**setinput(input);**

**}**

**void conversion()**

**{**

**inch=input%12;**

**feet=input/12;**

**}**

**int printinch()**

**{**

**return inch;**

**}**

**int printfeet()**

**{**

**return feet;**

**}**

**};**

**int main()**

**{ int input,inch,feet,response;**

**do{**

**cout<<"Enter the inches";**

**cin>>input;**

**inchToFeet i1(input);**

**i1.conversion();**

**feet=i1.printfeet();**

**inch=i1.printinch();**

**cout<<"height in Feet= "<<feet<<" "<<inch<<endl;**

**cout<<"DO YOU WANT ANOTHER(1/0) ";**

**cin>>response;**

**cout<<"\n\n\n";**

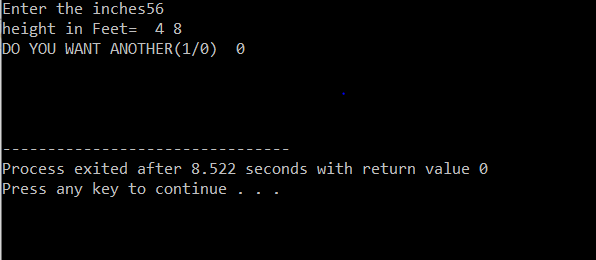
**}**

**while(response!=0);**

**return 0;**

**}**

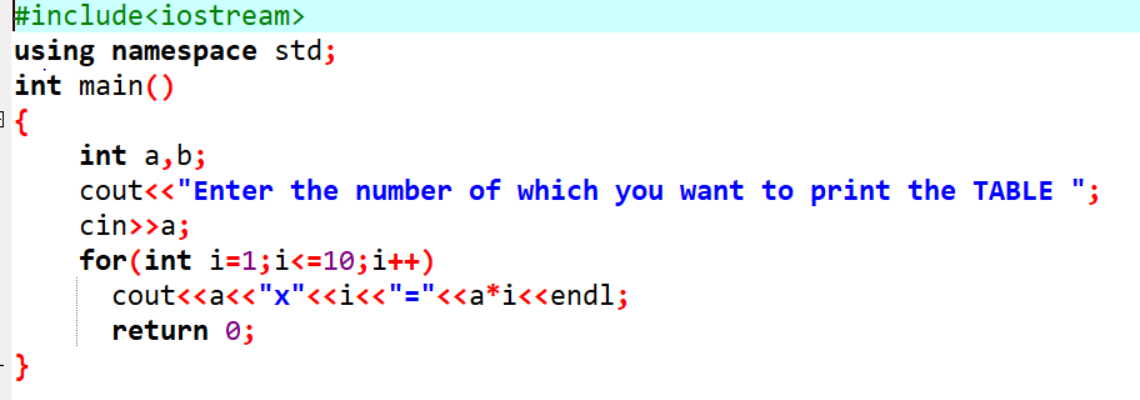
**OUTPUT-**



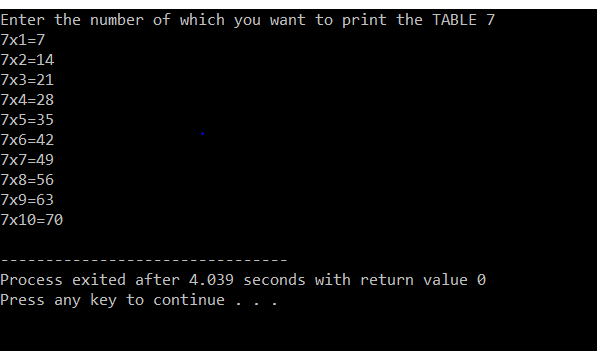
**EXPERIMENT 3.6**

**AIM-implement for loop**

**PROGRAM-**



**OUTPUT-**



**EXPERIMENT 4.1**

**AIM-Convert inch to feet using classes**

**PROGRAM-**

**#include<iostream>**

**using namespace std;**

**class inchToFeet{**

**int input;**

**int inch;**

**int feet;**

**public:**

**void setinput(int input)**

**{**

**this->input=input;**

**}**

**inchToFeet(int input)**

**{**

**setinput(input);**

**}**

**void conversion()**

**{**

**inch=input%12;**

**feet=input/12;**

**}**

**int printinch()**

**{**

**return inch;**

**}**

**int printfeet()**

**{**

**return feet;**

**}**

**};**

**int main()**

**{ int input,inch,feet,response;**

**do{**

**cout<<"Enter the inches";**

**cin>>input;**

**inchToFeet i1(input);**

**i1.conversion();**

**feet=i1.printfeet();**

**inch=i1.printinch();**

**cout<<"height in Feet= "<<feet<<" "<<inch<<endl;**

**cout<<"DO YOU WANT ANOTHER(1/0) ";**

**cin>>response;**

**cout<<"\n\n\n";**

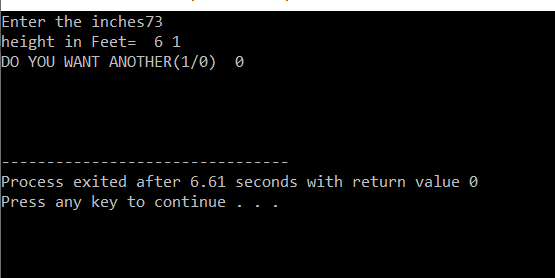
**}**

**while(response!=0);**

**return 0;**

**}**

**OTPUT**



**EXPERIMENT 4.2**

**AIM-**Consider rectangle, square, triangle and circle as 4 given shapes with a common area of 314 sq.cm. The length of a rectangle is 2cm and height of right-angled triangle is 4cm. Find the shape which has the smallest perimeter and largest perimeter

PROGRAM

**#include<iostream>**

**#include<math.h>**

**using namespace std;**

**class poly**

**{**

**float l,b,arect,perect,B,H,artri,pertri,s,arsq,persq,r,arcirc,percirc;**

**public:**

**float rect(float l1,float arect1)**

**{**

**l=l1;**

**arect=arect1;**

**b=arect/l;**

**perect=2\*(l+b);**

**return perect;**

**}**

**float tri(float H1,float artri1)**

**{**

**H=H1;**

**artri=artri1;**

**B=artri\*2/H;**

**pertri=2\*sqrt(pow(B,2)+pow(H,2))+B;**

**return pertri;**

**}**

**float sq(float arsq1)**

**{**

**arsq=arsq1;**

**s=sqrt(arsq);**

**persq=4\*s;**

**return persq;**

**}**

**float cir(float arcir1)**

**{**

**arcirc=arcir1;**

**r=sqrt(arcirc/3.14);**

**percirc=2\*3.14\*r;**

**return percirc;**

**}**

**}poly1,poly2;**

**int main()**

**{**

**float pr,pt,ps,pc,arr[4],a=INT\_MIN,b=INT\_MAX;**

**arr[0]=poly1.rect(2,314);**

**arr[1]=poly1.tri(4,314);**

**arr[2]=poly1.sq(314);**

**arr[3]=poly1.cir(314);**

**cout<<"PERIMETER OF RECTANGLE= "<<arr[0]<<"\n"<<"PERIMETER OF TRIANGLE= "<<arr[1]<<"\n"<<"PERIMETER OF SQUARE= "<<arr[2]<<"\n"<<"PERIMETER OF CIRCLE= "<<arr[3];**

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

**{**

**if(a<arr[i])**

**a=arr[i];**

**}**

**cout<<endl<<"LARGEST PERIMETER= "<<a<<endl;**

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

**{**

**if(b>arr[i])**

**b=arr[i];**

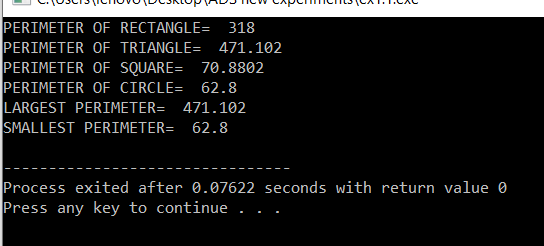
**}**

**cout<<"SMALLEST PERIMETER= "<<b<<endl;**

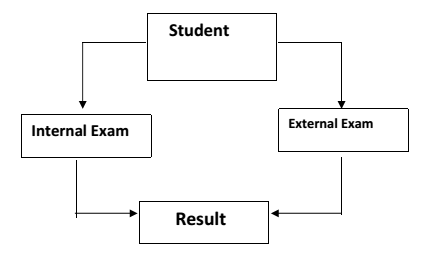
**return 0;**

**}**

**OUTPUT-**



**EXPERIMENT 5.1**

**AIM-**Implement Inheritance as shown in the diagram. Class Student has members functions student\_Details() and display\_Student\_Details() and name, roll no, and gender as data members. Internal\_Exam and External\_Exam classes have separate functions to get and display Internal and External marks for 6 subjects. Define Result class to evaluate and display the results suitably (40% Internal + 60% External).

PROGRAM-

#include<iostream>

#include<string>

using namespace std;

class student

{

private:

string name;

int roll\_no;

char gender;

public:

void student\_Details();

void display\_Student\_Details();

};

void student::student\_Details()

{

cout << "Enter student details!!" << endl;

cout << "NAME: ";

getline(cin,name);

cout << endl;

cout << "ROLL NO: ";

cin >> roll\_no;

cout << endl;

cout << "GENDER(M/F): ";

cin >> gender;

cout << endl;

}

void student::display\_Student\_Details()

{

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "STUDENT DETAILS:" << endl;

cout << "NAME: "<< name << endl;

cout << "ROLL NO: "<< roll\_no << endl;

cout << "GENDER: "<< gender << endl;

}

class Internal\_Exam :public student

{

protected:

int eng, maths, phys, chem, ADS, EVS;

public:

void getImarks();

float calcImarks();

void displayImarksr();

};

void Internal\_Exam::displayImarksr()

{

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "Your external exam marks are: " << endl << endl;

cout << " ENGLISH: " << eng << " ";

cout << " MATHS : " << maths << endl;

cout << " PHYSICS: " << phys << " ";

cout << " CHEMISTRY: " << chem << endl;

cout << " ADS : " << ADS << " ";

cout << " EVS : " << EVS << endl << endl;

cout << "Total marks in internals: " << calcImarks() << "%" << endl;

}

void Internal\_Exam::getImarks()

{

cout << "Enter the marks of English in internals: ";

cin >> eng;

cout << endl;

cout << "Enter the marks of maths in internals: ";

cin >> maths;

cout << endl;

cout << "Enter the marks of physics in internals: ";

cin >> phys;

cout << endl;

cout << "Enter the marks of Chemistry in internals: ";

cin >> chem;

cout << endl;

cout << "Enter the marks of ADS in internals: ";

cin >> ADS;

cout << endl;

cout << "Enter the marks of EVS in internals: ";

cin >> EVS;

cout << endl;

}

float Internal\_Exam::calcImarks()

{

float res;

res = eng + maths + phys + chem + ADS + EVS;

res = res / 6;

return res;

}

class External\_Exam :public student

{

protected:

int eng, maths, phys, chem, ADS, EVS;

public:

void getEmarks();

float calcEmarks();

void displayEmarkse();

};

void External\_Exam::displayEmarkse()

{

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "Your external exam marks are: " << endl << endl;

cout << " ENGLISH: " << eng << " ";

cout << " MATHS : " << maths << endl;

cout << " PHYSICS: " << phys << " ";

cout << " CHEMISTRY: " << chem << endl;

cout << " ADS : " << ADS << " ";

cout << " EVS : " << EVS << endl << endl;

cout << "Total marks in externals: " << calcEmarks() << "%" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

}

void External\_Exam::getEmarks()

{

cout << "Enter the marks of English in externals: ";

cin >> eng;

cout << endl;

cout << "Enter the marks of maths in externals: ";

cin >> maths;

cout << endl;

cout << "Enter the marks of physics in externals: ";

cin >> phys;

cout << endl;

cout << "Enter the marks of Chemistry in externals: ";

cin >> chem;

cout << endl;

cout << "Enter the marks of ADS in externals: ";

cin >> ADS;

cout << endl;

cout << "Enter the marks of EVS in externals: ";

cin >> EVS;

cout << endl;

}

float External\_Exam::calcEmarks()

{

float res2;

res2 = eng + maths + phys + chem + ADS + EVS;

res2 = res2 / 6;

return res2;

}

class result :public Internal\_Exam, public External\_Exam

{

private:

float f\_result;

public:

float calc\_fresult(float, float);

void dis\_fresult(float,float);

};

float result::calc\_fresult(float internal, float external)

{

f\_result = (internal \* 0.4 ) + (external \* 0.6);

return f\_result;

}

void result::dis\_fresult(float internal, float external)

{

cout << "Your internal total result: ";

cout << internal << "%" << endl;

cout << "Your external total result: ";

cout << external << "%" << endl;

cout << "Your total result :: ";

cout << f\_result << "%" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

}

int main()

{

float internal, external;

student main\_object;

Internal\_Exam int\_object;

External\_Exam ext\_object;

result final\_object;

main\_object.student\_Details();

int\_object.getImarks();

internal = int\_object.calcImarks();

ext\_object.getEmarks();

external = ext\_object.calcEmarks();

main\_object.display\_Student\_Details();

int\_object.displayImarksr();

ext\_object.displayEmarkse();

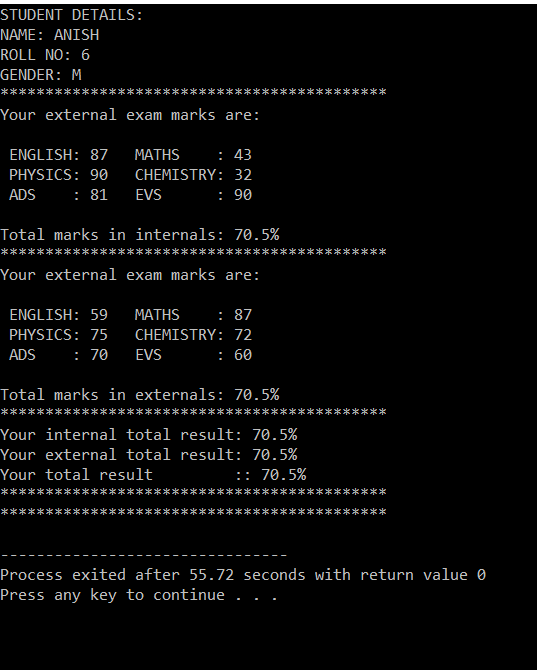
final\_object.calc\_fresult(internal,external);

final\_object.dis\_fresult(internal,external);

return 0;

}

OUTPUT-



**EXPERIMENT 5.2**

**AIM-implement any program on inheritance**

1. Base class ‘Temperature’ obtains temperature value in Kelvin (k) through its parameterized constructor, convert it into Celsius value (c) and prints it using the virtual function calculate(). Class ‘Temp’ inherits class ‘Temperature’ to obtain the temperature value in Celsius (c), converts it into Fahrenheit value (f) and prints it using the overridden function calculate().

**PROGRAM-**

**#include<iostream>**

**using namespace std;**

**class Temperature{**

**float k,c;**

**public:**

**Temperature(float k1=0)**

**{**

**k=k1;**

**}**

**virtual void calculate()**

**{**

**c=k-273;**

**cout<<"Temperature in celcius= "<<c<<endl;**

**}**

**void setc(float temp)**

**{**

**c=temp;**

**}**

**float gec()**

**{**

**return c;**

**}**

**};**

**class Temp:public Temperature**

**{**

**float f;**

**public:**

**void calculate()**

**{**

**f=(1.8f\*gec())+32;**

**cout<<"Temperatue in faraniet= "<<f<<endl;**

**}**

**Temp(float a)**

**{**

**setc(a);**

**}**

**};**

**int main()**

**{**

**Temp T1(1);**

**T1.calculate();**

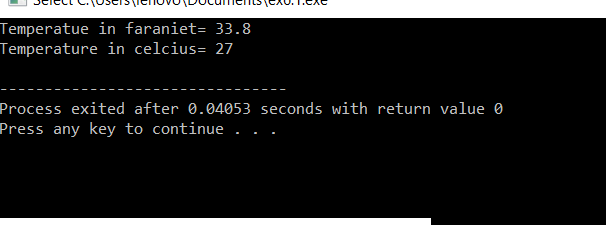
**Temperature t1(300);**

**t1.calculate();**

**return 0;**

**}**

**OUTPUT-**



**EXPERIMENT 6.1**

**AIM-Implement bubble sort**

**PROGRAM-**

**#include<iostream>**

**using namespace std;**

**int main()**

**{**

**int arr[10]={543,86,32,999,432,0,22,9,2,59};**

**int temp;**

**for(int i=8;i>0;i--)**

**{**

**for(int j=0;j<=i;j++)**

**{**

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

**{**

**temp=arr[j];**

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

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

**}**

**}**

**}**

**for(int i=0;i<=9;i++)**

**{**

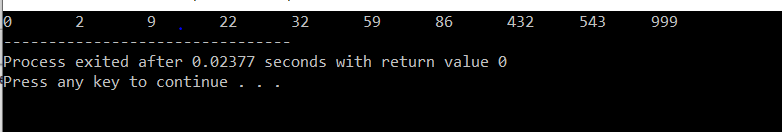
**cout<<arr[i]<<"\t";**

**}**

**return 0;**

**}**

**OUTPUT-**



**EXPERIMENT 6.2**

**AIM-implement exception handling**

**PROGRAM-**

**#include<iostream>**

**using namespace std;**

**float division(float a,float b,float ans);**

**int main()**

**{**

**float a,b,ans;**

**cout<<"Enter the nos you want to divide";**

**cin>>a>>b;**

**try**

**{**

**ans=division(a,b,ans);**

**cout<<"division of"<<a<<"and"<<b<<"="<<ans;**

**}**

**catch(int i)**

**{**

**cout<<"Division is not defined";**

**}**

**return 0;**

**}**

**float division(float a,float b,float ans)**

**{**

**if(b==0)**

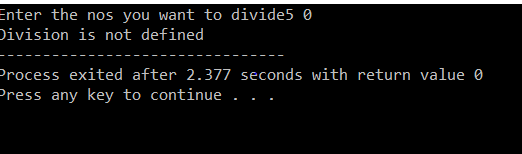
**throw(1);**

**ans=a/b;**

**return ans;**

**}**

**OUTPUT-**



**EXPERIMENT 7**

**AIM-impliment template in C++**

**PROGRAM-**

**#include<iostream>**

**using namespace std;**

**template<class T>**

**class calculator{**

**T a;**

**T b;**

**T ans;**

**public:**

**calculator(T a,T b)**

**{**

**this->a=a;**

**this->b=b;**

**}**

**T addision(){ans=a+b; return ans; }**

**T subtraction(){ans=a-b; return ans; }**

**T division(){ans=a/b; return ans; }**

**T multipication(){ans=a\*b; return ans; }**

**};**

**int main(){**

**calculator<int> o1(76,2);**

**cout<<"division="<<o1.addision()<<endl;**

**cout<<"subtraction="<<o1.subtraction()<<endl;**

**cout<<"multipication"<<o1.multipication()<<endl;**

**cout<<"divisin"<<o1.division()<<endl;**

**calculator<float> o2(7.8,5.2);**

**cout<<"division="<<o2.addision()<<endl;**

**cout<<"subtraction="<<o2.subtraction()<<endl;**

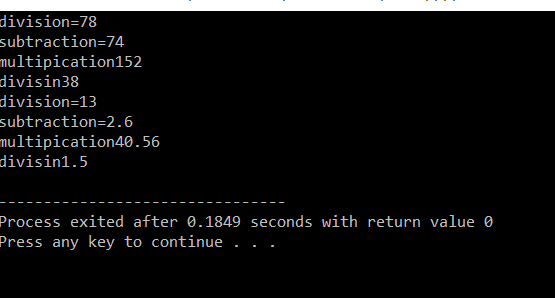
**cout<<"multipication"<<o2.multipication()<<endl;**

**cout<<"divisin"<<o2.division()<<endl;**

**return 0;**

**}**

**OUTPUT-**



**EXPERIMENT 8**

**AIM-implement hashing in C++**

**PROGRAM-**

**#include<iostream>**

**using namespace std;**

**int arr1[10]={7,89,43,90,54,43,77,33,10,0};**

**int arr2[20]={-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1};**

**void Hashing()**

**{**

**for(int i=0;i<=9;i++)**

**{**

**int h;**

**h=arr1[i]%20;**

**if(arr2[h]==-1)**

**arr2[h]=arr1[i];**

**else**

**{**

**for(int r=1;r<=9;r++)**

**{**

**h=(arr1[i]%20+r)%20;**

**if(arr2[h]==-1)**

**{arr2[h]=arr1[i];**

**break;**

**}**

**if(r==9)**

**cout<<"no place"<<endl;**

**}**

**}**

**}**

**}**

**void search(int a)**

**{**

**int i;**

**while(i!=-1)**

**{**

**static int r=0;**

**int h=(a%20+r)%20;**

**if(arr2[h]==a)**

**{**

**cout<<a<<" is present at index "<<h<<endl;**

**break;**

**}**

**r++;**

**i=arr2[h];**

**}**

**if(i==-1)**

**cout<<a<<"not found"<<endl;**

**}**

**void PrintTable()**

**{**

**for(int i=0;i<=19;i++)**

**cout<<arr2[i]<<" ";**

**cout<<endl;**

**}**

**int main()**

**{**

**int r,a,e;**

**cout<<"1. hashing\n"<<"2. print table\n"<<"3. search\n"<<"4. exit\n"<<endl;**

**do**

**{**

**cout<<"Enter your option ";**

**cin>>r;**

**switch(r)**

**{**

**case 1: Hashing();**

**break;**

**case 2: PrintTable();**

**break;**

**case 3: cout<<"Enter the number you want to search ";**

**cin>>a;**

**search(a);**

**break;**

**case 4: e=1;**

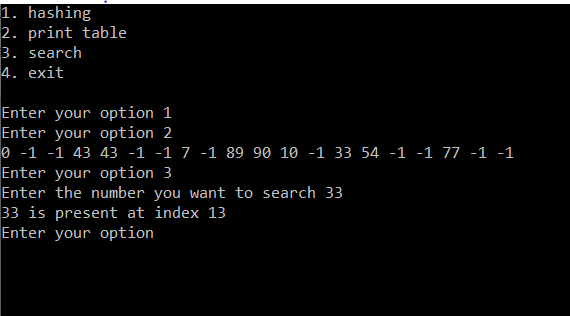
**}**

**}**

**while(e!=1);**

**}**

**OUTPUT**



**EXPERIMENT 8**

**AIM-Implement operator overloading in c++**

**PROGRAM-**

**#include<iostream>**

**using namespace std;**

**class complex{**

**int p;**

**int q;**

**public:**

**void setp(int p=0)**

**{**

**this->p=p;**

**}**

**void setq(int q=0)**

**{**

**this->q=q;**

**}**

**int getp()**

**{**

**return p;**

**}**

**int getq()**

**{**

**return q;**

**}**

**friend ostream& operator<< (ostream &o,complex ct)**

**{**

**o<<ct.p<<"+"<<"i"<<ct.q;**

**return o;**

**}**

**complex(int a=0,int b=0)**

**{**

**setp(a);**

**setq(b);**

**}**

**complex operator+(complex ct)**

**{**

**complex temp;**

**temp.p=p+ct.p;**

**temp.q=q+ct.q;**

**return temp;**

**}**

**friend complex operator%(complex ct1,complex ct2)**

**{**

**complex temp;**

**temp.p=ct1.p+ct2.p;**

**temp.q=ct1.q+ct2.q;**

**return temp;**

**}**

**};**

**int main()**

**{**

**complex c1(1,1),c2(88,9),c3,c4(6,7),c5(3,0);**

**cout<<c1<<endl;**

**cout<<c2<<endl;**

**c3=c1+c2;**

**cout<<c3<<endl;**

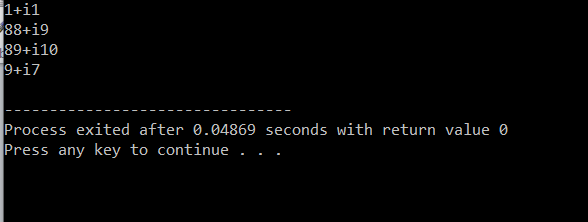
**c3=c4%c5;**

**cout<<c3<<endl;**

**return 0;**

**}**

**OUTPUT-**



**EXPERIMENT 9**

**AIM-Implement binary tree**

**PROGRAM-**

**#include<iostream>**

**using namespace std;**

**class node;**

**class queu{**

**int rear;**

**int front;**

**int size;**

**node \*\*p;**

**public:**

**queu(int size1=100)**

**{**

**size=size1;**

**p=new node\*[size];**

**front=rear=-1;**

**}**

**void enque(node \*x)**

**{**

**if(rear==size-1&&front==0||rear+1==front)**

**cout<<"queue is full\n";**

**else if(front==-1)**

**{**

**front=0;**

**rear=0;**

**p[rear]=x;**

**}**

**else**

**{**

**if(rear==size-1)**

**rear=0;**

**else**

**rear++;**

**p[rear]=x;**

**}**

**}**

**node\* deque()**

**{**

**node \*temp;**

**if(rear==-1)**

**cout<<"queue is empty\n";**

**else**

**{**

**temp=p[front];**

**p[front]=0;**

**if(front==size-1&&rear!=0)**

**front=0;**

**else if(rear==front)**

**{**

**p[front]=0;**

**front=-1;**

**rear=-1;**

**}**

**else**

**{**

**front++;**

**}**

**return temp;**

**}**

**}**

**int qisempty()**

**{**

**if(front==-1)**

**return 1;**

**else return 0;**

**}**

**};**

**class node**

**{**

**public:**

**node \*leftp;**

**int data;**

**node \*rightp;**

**};**

**node \*root;**

**class tree{**

**node \*r;**

**queu q1;**

**int y,z;**

**public:**

**tree()**

**{**

**root=NULL;**

**r=NULL;**

**int y;**

**node \*temp;**

**temp=new node;**

**q1.enque(temp);**

**cout<<"Enter the value of root node ";**

**cin>>temp->data;**

**root=temp;**

**}**

**void create()**

**{**

**while(!q1.qisempty())**

**{**

**r=q1.deque();**

**cout<<"Enter the left child of "<<r->data<<"\t";**

**cin>>y;**

**if(y!=-1)**

**{**

**node \*temp;**

**temp=new node;**

**temp->data=y;**

**q1.enque(temp);**

**r->leftp=temp;**

**}**

**else if(y==-1)**

**{**

**r->leftp=NULL;**

**}**

**cout<<"Enter the right child of "<<r->data<<"\t";**

**cin>>z;**

**if(z!=-1)**

**{**

**node \*temp;**

**temp=new node;**

**temp->data=z;**

**q1.enque(temp);**

**r->rightp=temp;**

**}**

**else if(z==-1)**

**{**

**r->rightp=NULL;**

**}**

**}**

**}**

**void preorder(node \*t)**

**{**

**if(t!=NULL)**

**{**

**cout<<t->data<<"\t";**

**preorder(t->leftp);**

**preorder(t->rightp);**

**}**

**}**

**void postorder(node \*t)**

**{**

**if(t!=NULL)**

**{**

**postorder(t->rightp);**

**postorder(t->leftp);**

**cout<<t->data<<"\t";**

**}**

**}**

**void inorder(node \*t)**

**{**

**if(t!=NULL)**

**{**

**inorder(t->leftp);**

**cout<<t->data<<"\t";**

**inorder(t->rightp);**

**}**

**}**

**node\* search(int x,node \*r=root)**

**{**

**if(r!=NULL)**

**{**

**if(r->data==x)**

**return r;**

**else if(r->data>x)**

**return search(x,r->leftp);**

**else if(r->data<x)**

**return search(x,r->rightp);**

**}**

**else {**

**cout<<x<<" not found\n";**

**return NULL;**

**}**

**}**

**node\* search2(int x,node \*r=root)**

**{**

**while(r!=NULL)**

**{**

**if(r->data==x)**

**return r;**

**else if(r->data<x)**

**{**

**r=r->rightp;**

**}**

**else if(r->data>x)**

**r=r->leftp;**

**}**

**if(r==NULL)**

**cout<<x<<" not found\n";**

**}**

**};**

**int main()**

**{**

**tree t1;**

**t1.create();**

**cout<<"PREORDER=";**

**t1.preorder(root);**

**cout<<"\nINORDER=";**

**t1.inorder(root);**

**cout<<"\nPOSTORDER=";**

**t1.postorder(root);**

**node \*add;**

**add=t1.search(37);**

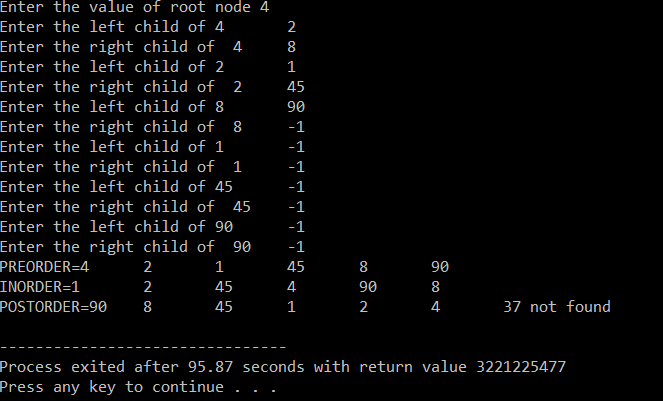
**cout<<add->data<<endl;**

**add=t1.search2(10);**

**cout<<add->data<<endl;**

**}**

**OUTPUT-**



**EXPERIMENT 10**

**AIM-implement graph in c++**

**PROGRAM-**

**#include<iostream>**

**using namespace std;**

**class queu{**

**int rear;**

**int front;**

**int size;**

**int\* p;**

**public:**

**queu(int size1=100)**

**{**

**size=size1;**

**p=new int[size];**

**front=rear=-1;**

**}**

**void enque(int x)**

**{**

**if(rear==size-1&&front==0||rear+1==front)**

**cout<<"queue is full\n";**

**else if(front==-1)**

**{**

**front=0;**

**rear=0;**

**p[rear]=x;**

**}**

**else**

**{**

**if(rear==size-1)**

**rear=0;**

**else**

**rear++;**

**p[rear]=x;**

**}**

**}**

**int deque()**

**{**

**int temp;**

**if(rear==-1)**

**cout<<"queue is empty\n";**

**else**

**{**

**temp=p[front];**

**p[front]=0;**

**if(front==size-1&&rear!=0)**

**front=0;**

**else if(rear==front)**

**{**

**p[front]=0;**

**front=-1;**

**rear=-1;**

**}**

**else**

**{**

**front++;**

**}**

**return temp;**

**}**

**}**

**int qisempty()**

**{**

**if(front==-1)**

**return 1;**

**else return 0;**

**}**

**};**

**queu q1;**

**class Graph**

**{**

**public:**

**int adjmatr[6][6]={{0,0,0,0,0,0},**

**{0,0,0,1,1,0},**

**{0,1,0,0,0,0},**

**{0,0,1,0,0,0},**

**{0,0,0,1,0,0},**

**{0,0,0,0,0,0}};**

**int checkB[6]={0,0,0,0,0,0};**

**void BFS1(int a)**

**{**

**cout<<a<<"\t";**

**checkB[a]=1;**

**q1.enque(a);**

**while(!q1.qisempty()){**

**a=q1.deque();**

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

**{**

**if(adjmatr[a][i]==1&&checkB[i]==0)**

**{**

**q1.enque(i);**

**checkB[i]=1;**

**cout<<i<<"\t";**

**}**

**}**

**}**

**}**

**int checkD[6]={0,0,0,0,0,0};**

**void DFS1(int a)**

**{**

**cout<<a<<"\t";**

**checkD[a]=1;**

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

**{**

**if(adjmatr[a][i]==1&&checkD[i]==0)**

**{**

**DFS1(i);**

**}**

**}**

**}**

**};**

**int main()**

**{**

**Graph g1;**

**g1.BFS1(1);**

**cout<<endl;**

**g1.DFS1(1);**

**}**

**OUTPUT**

