

oops practical file

GTBIT



**HARPREET SINGH**

**CSE-3**

**00776802719**

**Experiment-1**

**Aim:-** To find the roots of a given quadratic equation.

**Source Code:-**

#include<iostream>

#include<cmath>

using namespace std;

int main()

{

float a,b,c,discriminant,realpart,imaginarypart,x1,x2;

cout<<"\nEnter the coefficients of the quadratic equation: ";

cin>>a>>b>>c;

discriminant=(b\*b)-(4\*a\*c);

if(a==0)

{

cout<<"\nInvalid inputs\n";

}

else if(discriminant>0)

{

x1=(-b+sqrt(discriminant))/(2\*a);

x2=(-b-sqrt(discriminant))/(2\*a);

cout<<"\nThe roots are real and distinct: ";

cout<<"\nx1= "<<x1<<"\nx2= "<<x2;

}

else if(discriminant==0)

{

x1=-b/(2\*a);

cout<<"\nThe roots are real and equal:";

cout<<"\n x1=x2= "<<x1;

}

else

{

realpart=-b/(2\*a);

imaginarypart=sqrt(-discriminant)/(2\*a);

cout<<"\nThe roots are imaginary and distinct:";

cout<<"\nx1= "<<realpart<<" + "<<imaginarypart<<"i";

cout<<"\nx2= "<<realpart<<" - "<<imaginarypart<<"i";

}

cout<<"\n";

return 0;

}

**Algorithm:-**

**Start**

**Step 1:** Input the value of a, b, c.

**Step 2:** Calculate discriminant=(b\*b)-(4\*a\*c);

**Step 3:** if(a=0)

Output “Invalid Input”.

else If (d > 0)

Output "Roots are real and distinct” and calculate x1=(-b+sqrt(discriminant))/(2\*a),

x2=(-b-sqrt(discriminant))/(2\*a).

Output x1 and x2.

else if (d = 0)

Display "Roots are real and equal" and calculate x1 = x2= -b/(2\*a);

Output x1 and x2.

else

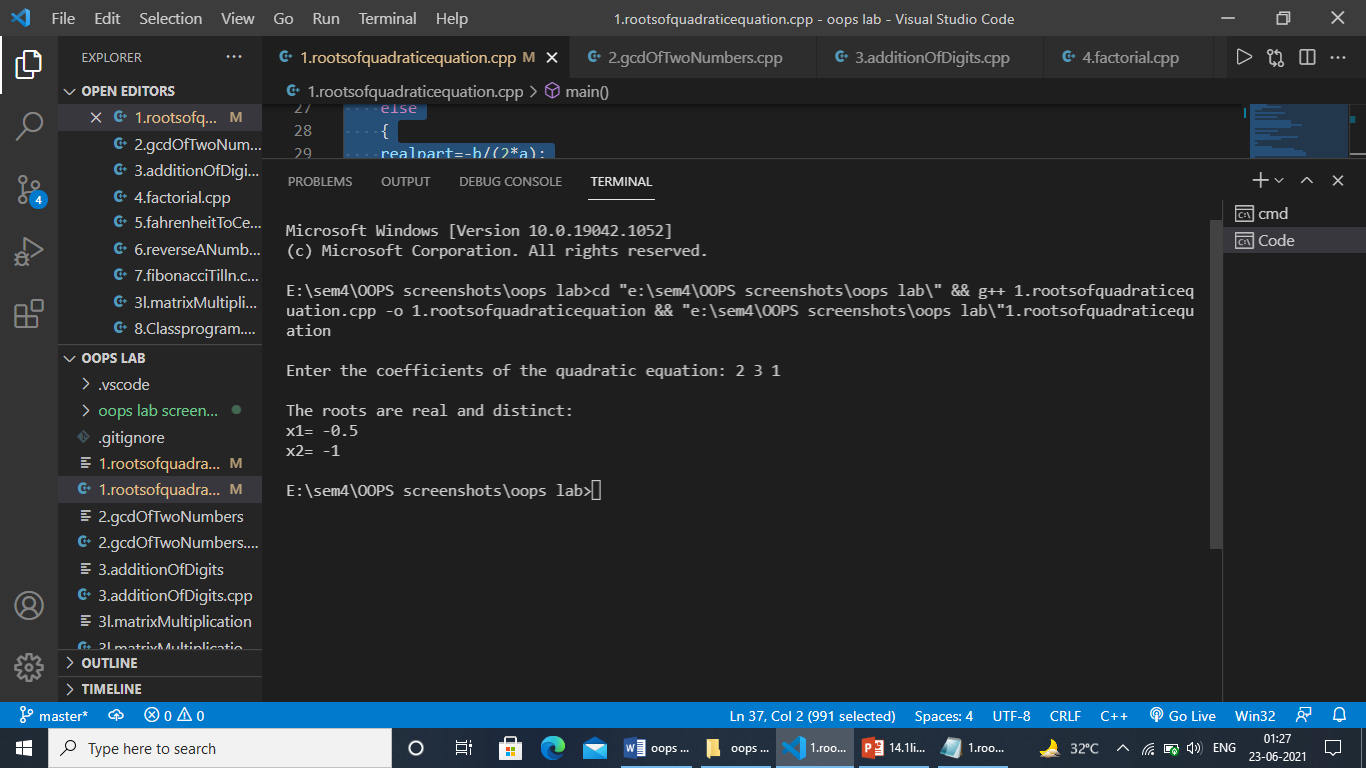
Display "Roots are Imaginary and distinct”, calculate x1=-b/(2\*a), x2 = sqrt(-discriminant)/(2\*a).

Output x1 and x2.

**Step 4:** End the algorithm.

**Stop**

**Output:-**



**Experiment-2**

**Aim:-** To find gcd and lcm of 2 nos.

**Source Code:-**

#include<iostream>

using namespace std;

void printGcdLcm(int num1,int num2)

{

int gcd,lcm,on1=num1,on2=num2;

while(num1%num2!=0)

{

int rem=num1%num2;

num1=num2;

num2=rem;

}

gcd=num2;

lcm=(on1\*on2)/gcd;

cout<<"\nGCD of "<<on1<<" and "<<on2<<" is: "<<gcd;

cout<<"\nLCM of "<<on1<<" and "<<on2<<" is: "<<lcm;

}

int main()

{

int num1,num2;

cout<<"Enter the two numbers: ";

cin>>num1>>num2;

printGcdLcm(num1,num2);

return 0;

}

**Algorithm:-**

**Start**

**Step 1:** read num1, num2, on1🡨num1 , on2🡨num2

**Step 2** while(num1%num2!=0)

* rem=num1%num2
* num1=num2
* num2=rem

End while

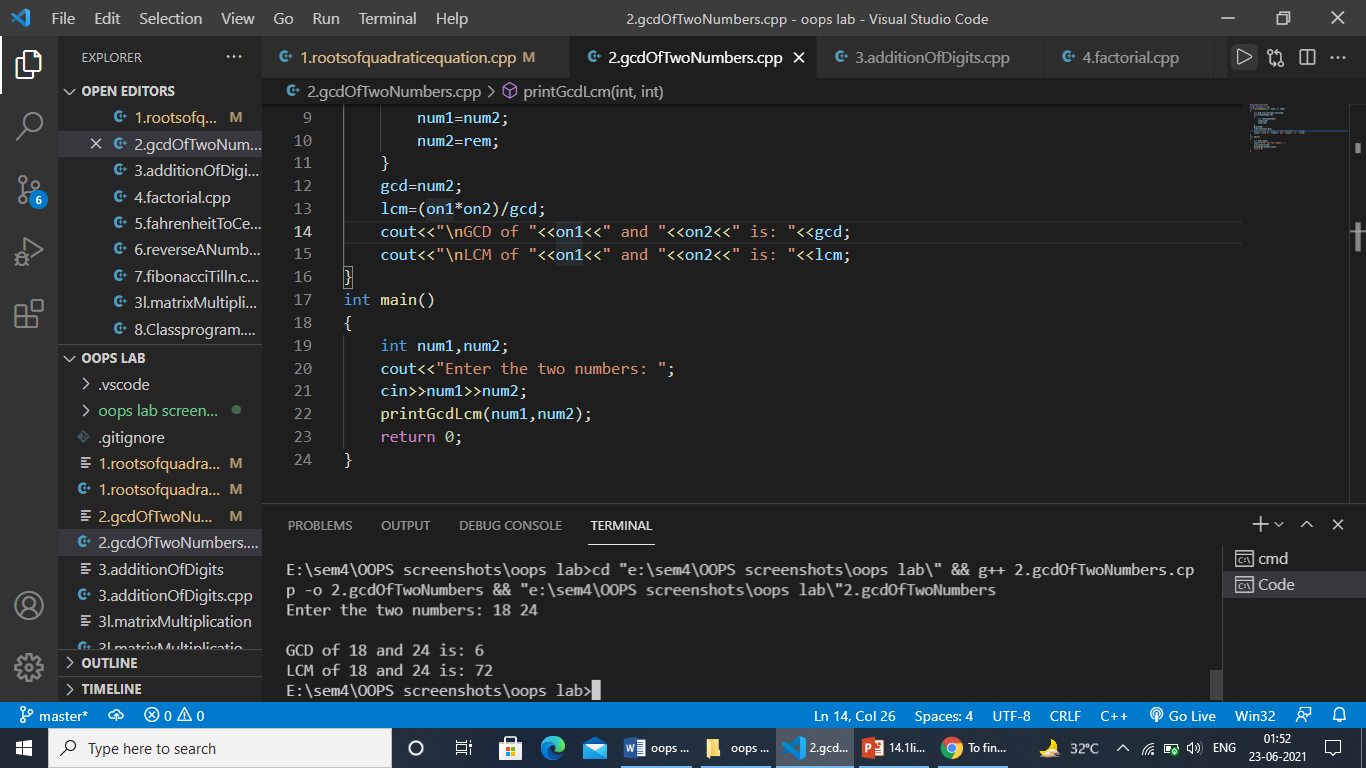
**step 3:** gcd 🡨num2

* + - Lcm 🡨(on1 \* on2) / gcd

**Step 4:** print gcd, lcm

**stop**

**Output:-**



**Experiment-3**

**Aim:-** To multiply two matrices using oop (using classes).

**Source Code:-**

#include <iostream>

using namespace std;

class Matrix

{

int a[10][10];

int b[10][10];

int c[10][10];

int i,j,k;

int m, n, p, q;

public:

Matrix()

{

cout<<"\nEnter the dimensions of matrix 1: ";

cin>>m>>n;

cout<<"\nEnter the dimensions of matrix 2: ";

cin>>p>>q;

}

void Mult();

void InputMatrix();

void OutputMatrix();

};

void Matrix::InputMatrix()

{

cout << "\nEnter the values for the first matrix\n";

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

{

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

{

cin >> a[i][j];

}

}

cout << "\nEnter the values for the second matrix\n";

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

{

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

{

cin >> b[i][j];

}

}

}

void Matrix::Mult()

{

if(n!=p)

{

cout<<"\nThe matrix multiplication is not possible\n";

return;

}

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

{

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

{

c[i][j]=0;

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

{

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

}

}

}

}

void Matrix::OutputMatrix()

{

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

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

{

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

{

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

}

cout << endl;

}

}

int main()

{

Matrix x;

x.InputMatrix();

x.Mult();

x.OutputMatrix();

return 0;

}

**Algorithm:-**

**Start**

**Step 1:**Start the Program.

**Step 2:**Enter the row and column of the first (a) matrix.

**Step 3:**Enter the row and column of the second (b) matrix.

**Step 4:**Enter the elements of the first (a) matrix.

**Step 5:**Enter the elements of the second (b) matrix.

**Step 6:**Set a loop up to row.

**Step 7:**Set an inner loop up to the column.

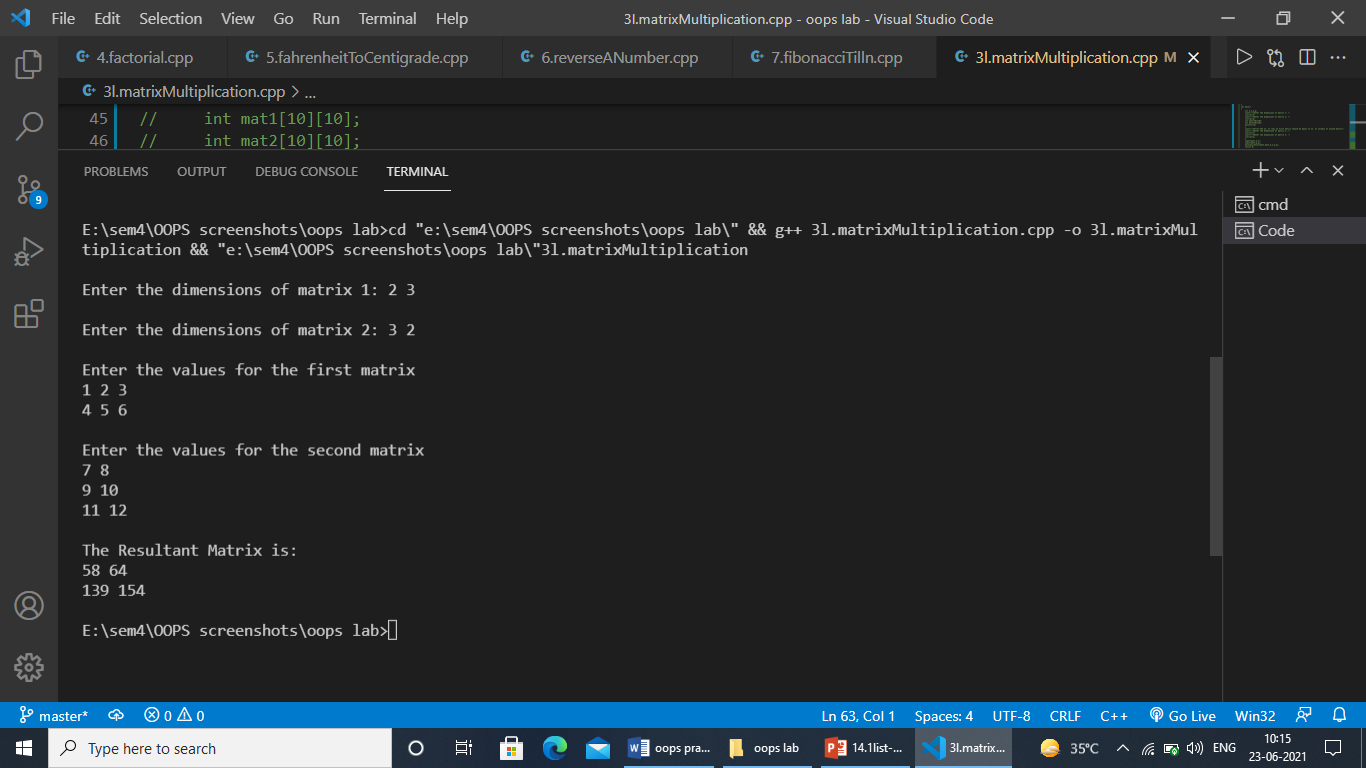
**Step 8:**Set another inner loop up to the column.

**Step 9:**Multiply the first (a) and second (b) matrix and store the element in the third matrix (c)

**Step 10:**Print the final matrix.

**Stop**

**Output:-**



**Experiment-4**

**Aim:-** Write a c++ program to find the greatest of two given numbers in two different classes using friend function.

**Source Code:-**

#include<iostream>

using namespace std;

class second;

class first

{

int x;

public:

void getx()

{

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

cin>>x;

}

friend void max(first,second);

};

class second

{

int y;

public:

void gety()

{

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

cin>>y;

}

friend void max(first,second);

};

void max(first a,second b)

{

if(a.x>b.y)

{

cout<<"\nGreater value is: "<<a.x;

}

else

{

cout<<"\nGreater value is: "<<b.y;

}

}

int main()

{

first a;

second b;

a.getx();

b.gety();

max(a,b);

return 0;

}

**Algorithm:-**

**Start**

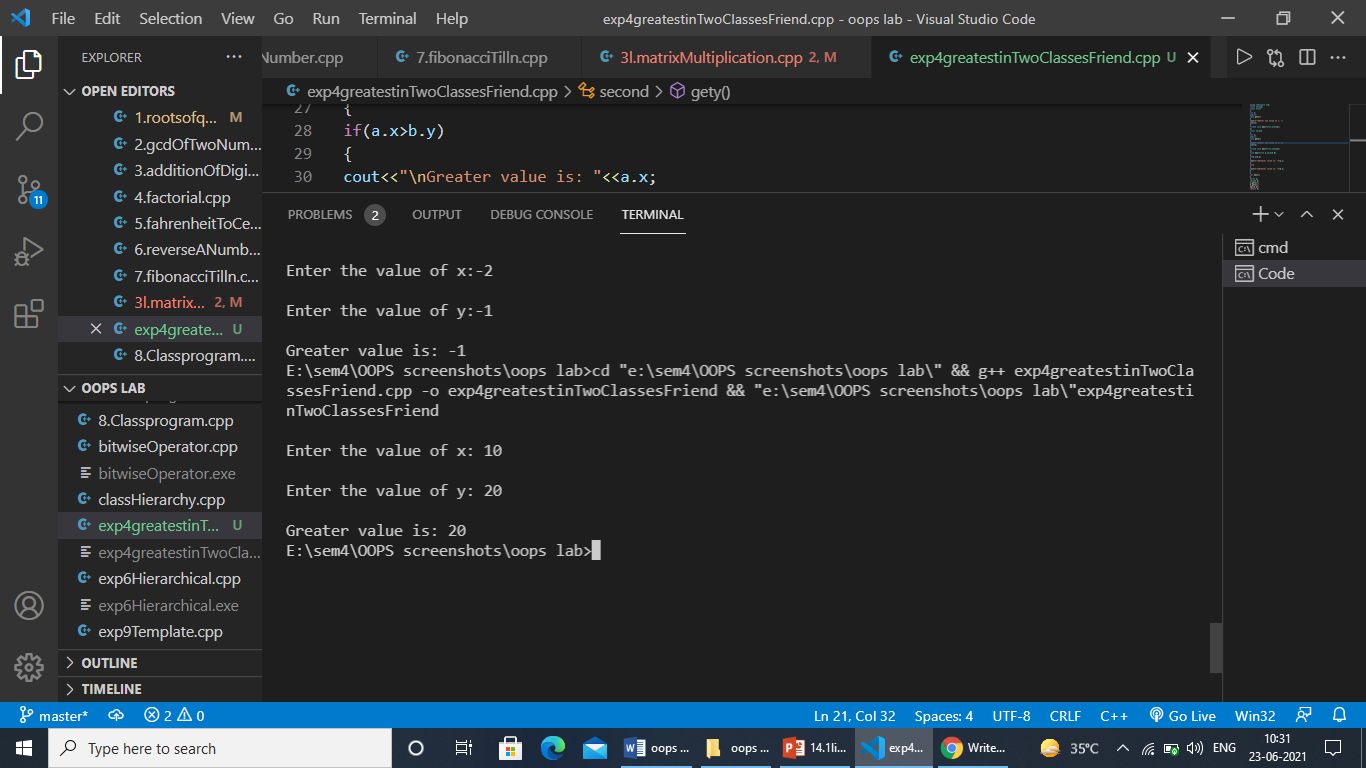
**Step1:** Enter the first number in first class.

**Step2:** Enter the second number in second class.

**Step3:** Compare the both the values in a friend function declared in both classes and print the greatest of two numbers.

**Stop**

**Output:-**



**Experiment-5**

**Aim:-**

**Source Code:-**

**Algorithm:-**

**Output:-**

**Experiment-2**

**Aim:-**

**Source Code:-**

**Algorithm:-**

**Output:-**

**Experiment-2**

**Aim:-**

**Source Code:-**

**Algorithm:-**

**Output:-**

**Experiment-2**

**Aim:-**

**Source Code:-**

**Algorithm:-**

**Output:-**

**Experiment-2**

**Aim:-**

**Source Code:-**

**Algorithm:-**

**Output:-**

**Experiment-2**

**Aim:-**

**Source Code:-**

**Algorithm:-**

**Output:-**

**Experiment-2**

**Aim:-**

**Source Code:-**

**Algorithm:-**

**Output:-**

**Experiment-2**

**Aim:-**

**Source Code:-**

**Algorithm:-**

**Output:-**