**C++ Questions**

**QUES-1 WAP to compute the sum of the first n terms of the following series: //S = 1 - 1 / (2 ^ 2) + 1 / (3 ^ 3) - ... 1 / (n ^ n) .**

**=>**#include<iostream>

#include<math.h>

using namespace std;

int main(int argument,char\* a[])

{

int n;

if(argument==2)

{

    //atoi is a predefined function used to convert a string value to an integer value.

    n=atoi(a[1]);

}

else if(argument!=2)

{

    cout<<"Enter the value of n: ";

    cin>>n;

}

float sum=0.0;

float i=1;

while (i<=n)

{

    sum=sum+pow(-1,i+1)\*(1.0/(i\*i));

    i=i+1;

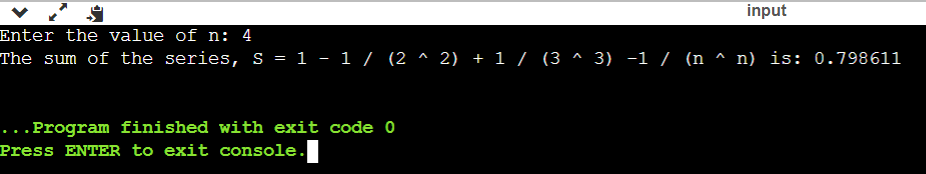
}

cout<<"The sum of the series, S = 1 - 1 / (2 ^ 2) + 1 / (3 ^ 3) -1 / (n ^ n) is: "<<sum<<endl;

    return 0;

}

**OUTPUT:-**



**QUES 2 WAP to remove the duplicates from an array.**

**=>**#include<iostream>

using namespace std;

int main()

{

int a[30];

int i,j,n=5,k;

cout<<"Enter elements of an array: ";

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

cin>>a[i];

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

{

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

{

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

{

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

{

a[k]=a[k+1];

}

--n;

}

else

++j;

}

}

cout<<"Resultant array without duplicants: ";

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

{

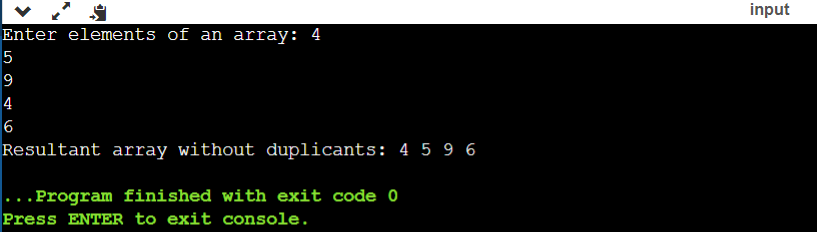
cout<<a[i]<<" ";

}

return 0;

}

**OUTPUT:-**



**QUES -3 Write a program that prints a table indicating the number of occurrences of each alphabet in the text entered as command line arguments.**

**=>**#include<iostream>

#include<string.h>

using namespace std;

int main(int argc, char \*argv[])

{

    int i;

    int count=0;

    cout<<"Parameters are: ";

    for (int i=0;i<argc; i++)

    cout<<argv[i]<<" ";

    for(char k= 'a'; k<= 'z'; k++)

    {

        for(i=0; i<argc; i++)

        {

            for(int j=0; argv[i][j]!='\0';j++)

            {

                argv[i][j]=tolower(argv[i][j]);

                if(argv[i][j] ==k)

                count++;

            }

        }

        if (count>0)

        cout<<"\n"<<k<<" occurs "<<count<<" times.";

        count=0;

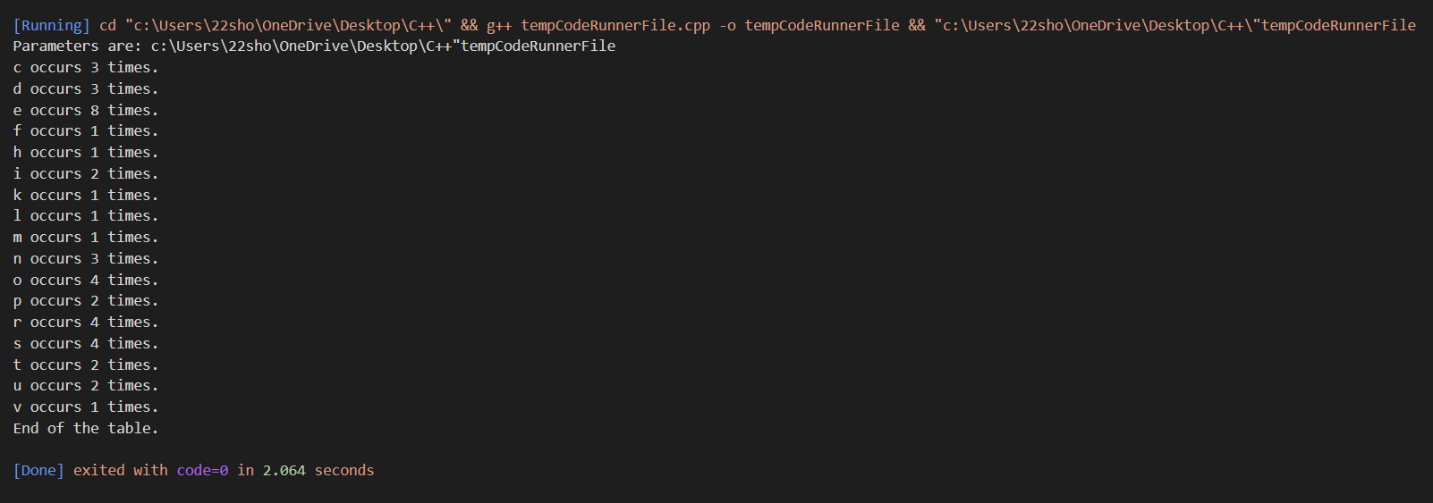
    }

    cout<<"\nEnd of the table."<<endl;

    return 0;

}

**OUTPUT:-**



**QUES 4-Write a menu driven program to perform following operations on strings (without using inbuilt string functions):**

**a) Show address of each character in string.**

**b) Concatenate two strings.**

**c) Compare two strings.**

**d) Calculate length of the string (use pointers).**

**e) Convert all lowercase characters to uppercase.**

**f) Reverse the string.**

**=>**#include <iostream>

#include<cstdlib>

#include<string.h>

#include<iomanip>

using namespace std;

void printaddress(char a[])

{

    void\* p;

    for(int i=0;i<strlen(a);i++)

    {

        p= &a[i];

        cout<<"Character"<<setw(2)<<a[i]<<setw(9)<<" address is: "<<p<<endl;

    }

}

int length(char\* p)

{

    int i=0;

    while(\*p)

    {

        i++;

        p=p+1;

    }

    return i;

}

void concatenate(char\* p, char\* p1)

{

    while(\*p!='\0')

    p++;

    while(\*p1!='\0')

    {

        \*p=\*p1;

        p++;

        p1++;

    }

    \*p='\0';

}

void relationalOperation(char s1[],char s2[])

{

}

void reverse(char p[])

{

    char temp;

    for(int i=0,j=length(p)-1;i<length(p)/2;i++,j--)

    {

        temp=p[i];

        p[i]=p[j];

        p[j]=temp;

    }

}

void case\_changer(char s[])

{

    int i;

    for(i=0;i<strlen(s);i++)

    if(islower(s[i]))

    {

        s[i]=s[i]-32;

    }

    return;

}

int main()

{

    char s1[20],s2[20];

    int count,x1,p1,p2;

    cout<<"1. Input a string: "<<endl;

    cout<<"2. Print address of each character of the string: "<<endl;

    cout<<"3. Concatenation of two strings: "<<endl;

    cout<<"4. Comparison of two strings: "<<endl;

    cout<<"5. Length of string: "<<endl;

    cout<<"6. Conversion of all lowercase characters to uppercase : "<<endl;

    cout<<"7. Reverse the string: "<<endl;

    char ch='y';

    while(ch=='y')

    {

        cout<<"Enter your choice: ";

        cin>>x1;

        switch(x1)

        {

            case 1: cout<<"Enter the first string: ";

            cin>>s1;

            cout<<"Enter the second string: ";

            cin>>s2;

            break;

            case 2: printaddress(s1);

            printaddress(s2);

            break;

            case 3: concatenate(s1,s2);

            cout<<"Concatenation of two string are: "<<s1<<endl;

            break;

            case 4: relationalOperation(s1,s2);

            if(s1==s2)

            {

            cout<<s1<<" is equal to "<<s2<<endl;

            }

            else

            cout<<s1<<" is not equal to "<<s2<<endl;

            break;

            case 5: p1=length(s1);

            cout<<"Length of the first string is: "<<p1<<endl;

            p2=length(s2);

            cout<<"Length of the second string is: "<<p2<<endl;

            break;

            case 6: case\_changer(s1);

            cout<<"Uppercase of the first string is: "<<s1<<endl;

            case\_changer(s2);

            cout<<"Uppercase of the second string is: "<<s2<<endl;

            break;

            case 7: reverse(s1);

            cout<<"Reverse of the first string is: "<<s1<<endl;

            reverse(s2);

            cout<<"Reverse of the second string is: "<<s2<<endl;

            break;

            default: cout<<"Invalid choice!! "<<endl;

        }

        cout<<"Do you want to continue?"<<" ";

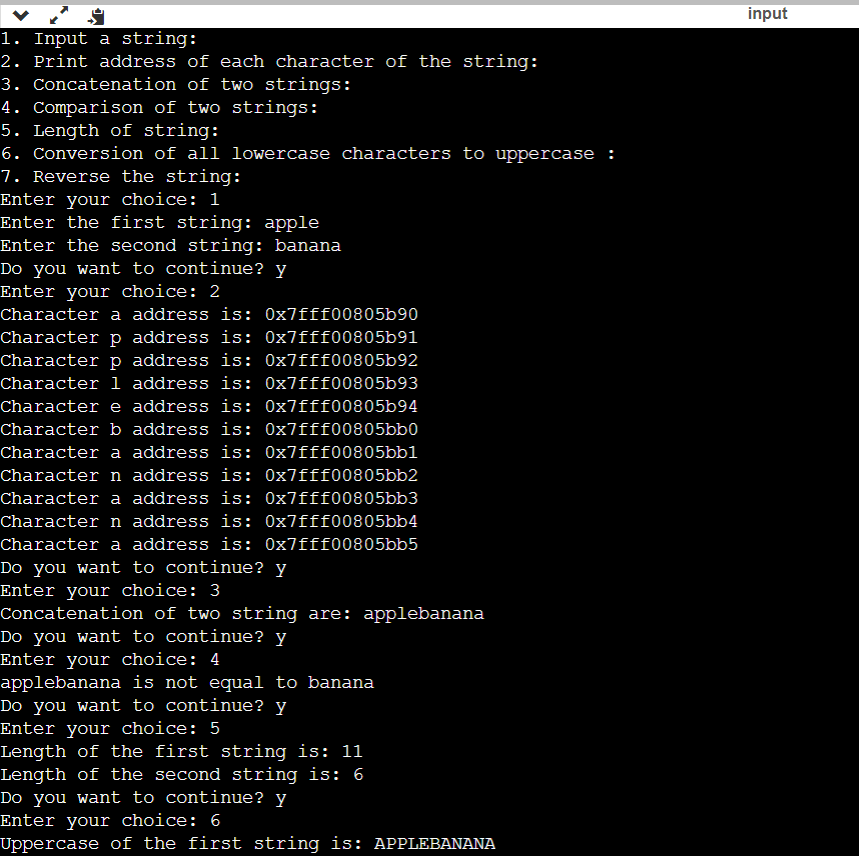
        cin>>ch;

    }

    return 0;

}

**OUTPUT:-**



**QUES -5 Write a program to merge two ordered arrays to get a single ordered array.**

**=>**#include <iostream>

using namespace std;

int main()

{

int a1[5], a2[5], a3[10];

int size1 = 5, size2 = 5, i,t;

cout<<"Enter elements for first array: ";

for(i=0;i<size1;i++)

{

cin>>a1[i];

a3[i] = a1[i];

}

t=i;

cout<<"Enter elements for second array: ";

for(i=0;i<size2;i++)

{

cin>>a2[i];

a3[t] = a2[i];

t++;

}

cout<<"Combined single ordered array is: ";

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

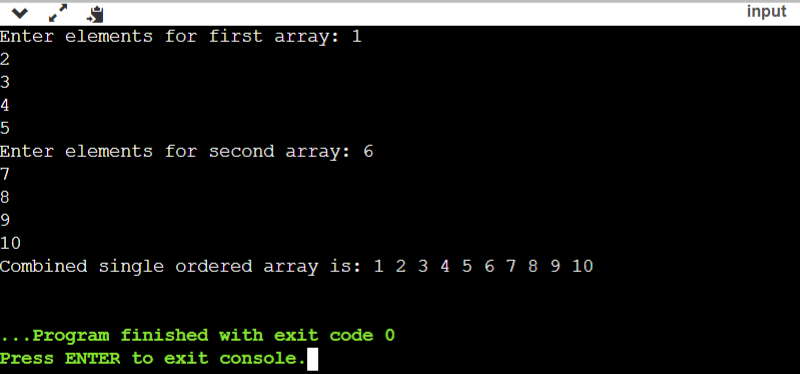
cout<<a3[i]<<" ";

cout<<endl;

return 0;

}

**OUTPUT:-**



**QUES 6 - Write program to each a given element in a set of N numbers using Binary search: (i) with recursion.**

**=>**(i) #include<iostream>

using namespace std;

int binarySearch(int a[],int l,int h, int n)

{

if(l<=h)

{

int m=(l+h)/2;

if(a[m]==n)

{

    return m;

}

else if(a[m] <n)

{

    return binarySearch(a,m+1,h,n);

}

else

{

    return binarySearch(a,l,m-1,n);

}

}

return -1;

}

int main()

{

    int a[]={1,3,5,8,10};

    int n=5,l,h,m, index;

    l=0;

    h=(sizeof(a)/sizeof(a[0]))-1;

    index= binarySearch(a,l,h,n);

    if(index==-1)

    {

        cout<<"Number not found. "<<endl;

    }

    else

    {

        cout<<"Number found at index: "<<index<<endl;

    }

    return 0;

}

**OUTPUT:-**



(ii)

#include<iostream>

using namespace std;

int main()

{

int a[]={2,4,6,7,8,9};

int n=9,l,h,m,mid;

l=0;

h=(sizeof(a)/ sizeof(a[0]))-1;

while(l<=h)

{

   m=l+h/2;

   if(a[m]==n)

   {

    break;

   }

  else if(a[m]<n)

  {

    l=m+1;

  }

  else

  {

   h=m-1;

   }

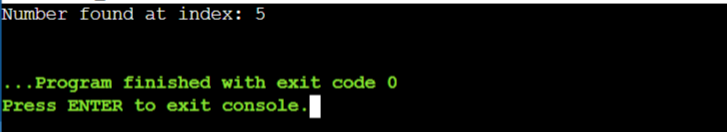
}

cout<<"Number found at index: "<<m<<endl;

   return 0;

}

**OUTPUT:-**



**QUES 7 - Write a program to calculate GCD of two numbers (i) with recursion (ii) without recursion. (i)with recursion.**

**=>**i) #include <iostream>

using namespace std;

int gcd(int a,int b)

{

    if(a == 0 || b == 0)

    return 0;

    if (a>b)

    return gcd(a-b,b);

    return gcd(a,b-a);

}

int main()

{

    int a,b;

    cout<<"Enter two numbers: ";

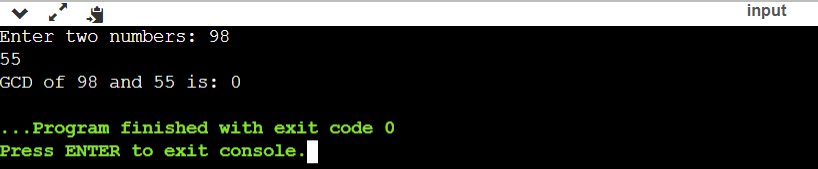
    cin>>a>>b;

    cout<<"GCD of "<<a<<" and "<<b<<" is: "<<gcd(a,b);

    return 0;

}

**OUTPUT:-**



ii) #include <iostream>

using namespace std;

int main()

{

    int num1, num2, gcd;

    cout << " Input the first number: ";

    cin >> num1;

    cout << " Input the second number: ";

    cin >> num2;

    for (int i = 1; i <= num1 && i <= num2; i++)

    {

        if (num1 % i == 0 && num2 % i == 0)

        {

            gcd = i;

        }

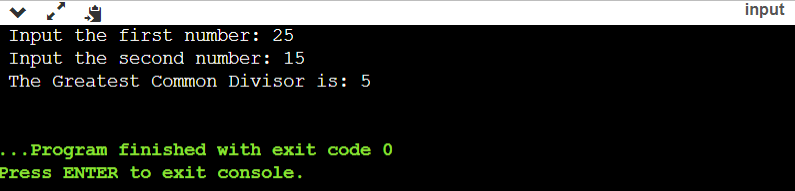
    }

    cout << " The Greatest Common Divisor is: " << gcd << endl;

    return 0;

}

**OUTPUT:-**



**QUES-8 Create Matrix class. Write a menu-driven program to perform following Matrix operations: a) Sum b) Product c) Transpose.**

**=>**#include<iostream>

#include<iomanip>

using namespace std;

class Matrix

{

private:

int a[6][6];

int row, col;

public:

Matrix(int row=5, int col=5){

(\*this).row=row;

(\*this).col=col;

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

{

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

a[i][j]=0;

}

}

void input();

void display();

Matrix matrixadd(Matrix o1);

Matrix matrixsub(Matrix o1);

Matrix matrixmultiply(Matrix o1);

Matrix matrixtranspose();

};

void Matrix::input()

{

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

{

cout<<"Enter the elements of row "<<(i+1)<<" : ";

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

cin>>a[i][j];

}

}

void Matrix::display()

{

cout<<endl;for(int i=0; i<row; i++)

{

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

cout<<setw(3)<<a[i][j];

cout<<endl;

}

cout<<endl;

}

Matrix Matrix::matrixadd(Matrix o1)

{

Matrix o3;

o3.row=row;

o3.col=col;

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

{

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

o3.a[i][j]=a[i][j]+o1.a[i][j];

}

return o3;

}

Matrix Matrix::matrixsub(Matrix o1)

{

Matrix o3;

o3.row=row;

o3.col=col;

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

{

for(int j=0; j<col; j++ )o3.a[i][j]=a[i][j]-o1.a[i][j];

}

return o3;

}

Matrix Matrix::matrixtranspose()

{

Matrix temp;

temp.row=col;

temp.col=row;

for(int i=0; i<temp.row; i++)

{

for(int j=0; j<temp.col; j++ )

temp.a[i][j]=a[j][i];

}

return temp;

}

Matrix Matrix::matrixmultiply(Matrix o1)

{

Matrix o7;

o7.row=row;

o7.col=o1.col;

if(col==o1.row)

{

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

{

for(int j=0; j<o1.col; j++ )

{

o7.a[i][j]=0;for(int k=0;k<col;k++)

o7.a[i][j]=o7.a[i][j]+a[i][k]\*o1.a[k][j];

}

}

}

return o7;

}

int main()

{

Matrix o1(4,3) ,o2(4,3), o3(4,3), o4(4,3), o5(3,4), o6(3,4), o7(3,3), o8(3,4);

cout<<"Enter the elements of first matrix:- "<<endl;

o1.input();

o1.display();

cout<<"\nEnter the elements of second matrix:- "<<endl;

o2.input();

o2.display();

cout<<"\nEnter the elements of third matrix:- "<<endl;

o8.input();

o8.display();

cout<<"\nMultiplication of the two matrix are: ";

o7=o1.matrixmultiply(o8);

o7.display();

cout<<"\nAddition of the two matrix are: ";

o3=o1.matrixadd(o2);

o3.display();

cout<<"\nSubtraction of the two matrix are: "<<endl;

o4=o1.matrixsub(o2);

o4.display();

cout<<"\nTranspose of the first matrix is: "<<endl;o5=o1.matrixtranspose();

o5.display();

cout<<"\nTranspose of the second matrix is: "<<endl;

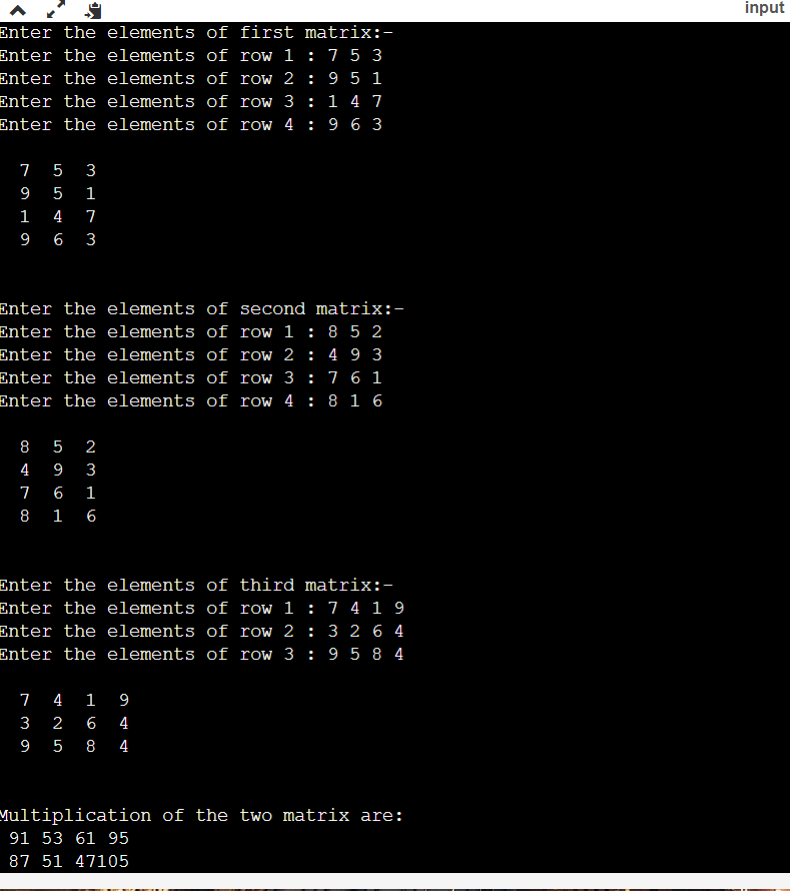
o6=o2.matrixtranspose();

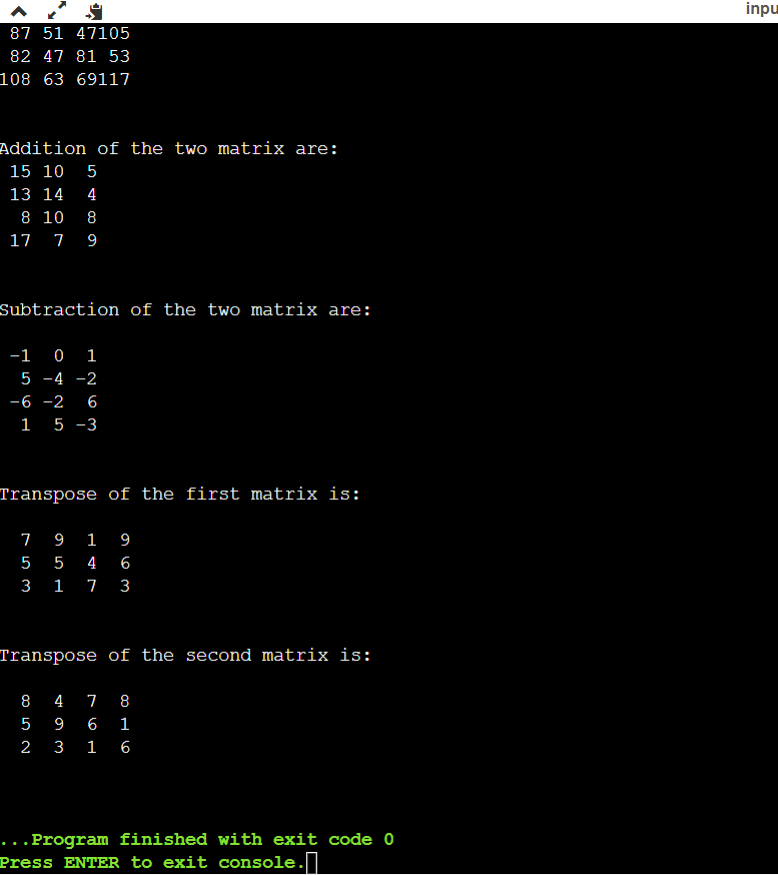
o6.display();

return 0;

}

**OUTPUT:-**





**QUES-9 Define a class Person having name as a data member. Inherit two classes Student and Employee from Person. Student has additional attributes as course, marks and year and Employee has department and salary. Write display() method in all the three classes to display the corresponding attributes. Provide the necessary methods to show runtime polymorphism.**

**=>**#include <iostream>

#include<cstring>

using namespace std;

class Person

{

  public:

  string name;

  void get();

  void display();

  };

  void Person::get()

  {

    cout<<"Enter name of the employee: ";

    getline(cin,name);

  }

  void Person::display()

  {

    cout<<"Name of the employee is: ";

    cout<<name;

  }

  class Student: public Person

  {

    public:

    int cls;

    int rollno;

    public:

    void get();

    void display();

  };

void Student::get()

{

  cout<<"Enter class of the employee: ";

  cin>>cls;

cout<<"Enter Roll number of employee: ";

  cin>>rollno;

}

void Student::display()

{

  cout<<"\nClass of the employee is: "<<cls<<endl;

cout<<"Roll number of the employee is: "<<rollno<<endl;

}

class Employee: public Person

{

  public:

  int salary;

  int id;

  public:

   void get();

    void display();

};

void Employee::get()

{

  cout<<"Enter salary of the employee: ";

  cin>>salary;

cout<<"Enter id of the employee: ";

  cin>>id;

}

void Employee::display()

{

cout<<"\nSalary of the employee is: "<<salary<<endl;

cout<<"Id of the employee is: "<<id<<endl;

}

int main()

{

Person p1;

Employee e1;

Student s1;

p1.get();

e1.get();

e1.display();

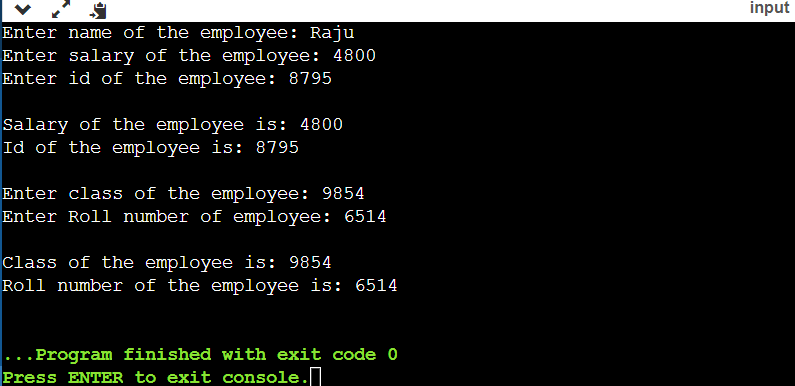
cout<<endl;

s1.get();

s1.display();

}

**OUTPUT:-**



**QUES-10 Create a class Triangle. Include overloaded functions for calculating area. Overload assignment operator and equality operator.**

**=>**#include<iostream>

#include<math.h>

using namespace std;

class Triangle

{

private:

float area;

public:

Triangle()

{

area=0;

}

void triangleArea(float a);

void triangleArea(float a,float b);

void triangleArea(float a,float b,float c);

Triangle operator=(Triangle o2);

int operator==(Triangle o2);

void display()

{

cout<<"The area of triangle is: "<<area<<endl;

}

};

void Triangle::triangleArea(float a)

{

area= (sqrt(3)/4)\*a\*a;

cout<<"Area of an equilateral triangle is: "<<area<<endl;

}

void Triangle::triangleArea(float a, float b)

{

area= 0.5\*a\*b;

cout<<"Area of an isosceles triangle is: "<<area<<endl;

}

void Triangle::triangleArea(float a, float b, float c)

{

float s= (a+b+c)/2;

float m=(s-a)\*(s-b)\*(s-c);

area= sqrt(m);

cout<<"Area of a scalene triangle is: "<<area<<endl;

}

Triangle Triangle::operator=(Triangle o2)

{

area=o2.area;

return o2;

}

int Triangle::operator==(Triangle o2)

{

if(area==o2.area)

return 1;

else

return 0;

}

int main()

{

Triangle o1,o2,o3,o4;

o1.triangleArea(6);

o2.triangleArea(8.7,5.2);

o3.triangleArea(9,7,2);

o4=o1;

o4.display();

o1.display();

if(o1==o4)

cout<<"\nArea is equal. "<<endl;

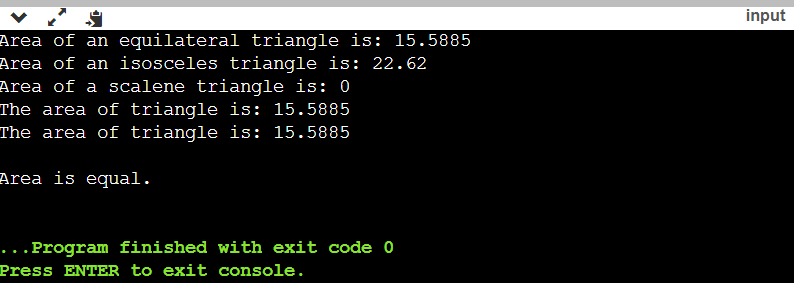
else

cout<<"\nArea is not equal. "<<endl;

return 0;

}

**OUTPUT:-**



**QUES 11. Write a program to read two numbers p and q. If q is 0 then throw an exception else display the result of p/q.**

**=>**#include<iostream>

using namespace std;

int main()

{

double a,b,quotient;

cout<<"Enter the numerator: ";

cin>>a;

cout<<"Enter the denominator: ";

cin>>b;

try

{

if(b==0)

throw "Attempt to divide by zero";

quotient=a/b;

cout<<"The quotient is: "<<quotient<<endl;

}

catch(const char\* p)

{

cout<<p<<endl;

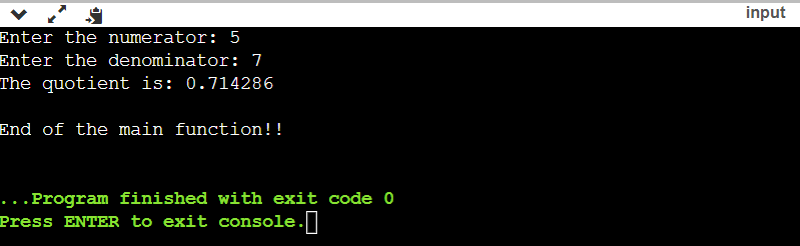
}

cout<<"\nEnd of the main function!!"<<endl;

return 0;

}

**OUTPUT:-**



**QUES 12. Rewrite Matrix class of Q8 with exception handling. Exceptions should be thrown by the functions if matrices passed to them are incompatible and handled by main () function.**

**=>**#include <iostream>

#include <iomanip>

using namespace std;

class matrix

{

  int a[5][5];

  int row, col;

  public:

  void input ();

  void display ();

  matrix operator \* (matrix o1);

  matrix operator + (matrix o1);

  matrix operator - (matrix o1);

  matrix transpose ();

    matrix (int row = 5, int col = 5)

  {

    (\*this).row = row;

    (\*this).col = col;

    for (int i = 0; i < row; i++)

    for (int j = 0; j < col; j++)

    a[i][j] = 0;

  }

};

void matrix::input ()

{

  for (int i = 0; i < row; i++)

    {

      cout << " Enter the elements of row "<< (i + 1)<<": ";

      for (int j = 0; j < col; j++)

      cin >> a[i][j];

    }

}

matrix matrix::operator \* (matrix o1)

{

  cout <<"Multiplication of matrix 1st and 2nd\n";

  matrix o3;

  if (col == o1.row)

    {

      o3.row = row;

      o3.col = o1.col;

      for (int i = 0; i < row; i++)

      {

        for (int j = 0; j < o1.col; j++)

         {

           o3.a[i][j] = 0;

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

           o3.a[i][j] = o3.a[i][j] + a[i][k] \* o1.a[k][j];

         }

      }

    }

  return o3;

}

matrix matrix::transpose ()

{

  matrix temp;

  temp.row = col;

  temp.col = row;

  for (int i = 0; i < temp.row; i++)

    for (int j = 0; j < temp.col; j++)

      temp.a[i][j] = a[j][i];

  return temp;

}

matrix matrix::operator + (matrix o1)

{

  cout <<"Addition of 1st and 3rd matrix\n";

  matrix o6;

  o6.row = row;

  o6.col = o1.col;

  if (row != o1.row || col != o1.col)

    throw "\n Incompatible matrix\n";

    for (int i = 0; i < row; i++)

    for (int j = 0; j < o1.col; j++)

    o6.a[i][j] = a[i][j] + o1.a[i][j];

    return o6;

}

matrix matrix::operator - (matrix o1)

{

  cout <<"Subtraction of 1st and 3rd matrix\n ";

  matrix o6;

  o6.row = row;

  o6.col = o1.col;

  if (row != o1.row || col != o1.col)

    throw "\nIncompatible matrix\n";

    for (int i = 0; i < row; i++)

    for (int j = 0; j < o1.col; j++)

    o6.a[i][j] = a[i][j] - o1.a[i][j];

    return o6;

}

void matrix::display ()

{

  cout <<"\n\*\*\*\*\* Your matrix is \*\*\*\*\*"<< endl;

  for (int i = 0; i < row; i++)

    {

      for (int j = 0; j < col; j++)

      cout << setw (3) << a[i][j];

      cout << endl;

    }

}

int main ()

{

  int c, x;

  char ch = 'y';

  matrix o1 (3, 4), o2 (4, 3), o3 (3, 3), o4 (4, 3), o5 (3, 4), o6 (3, 4);

  cout <<  "\n\*\*\*MATRIX CLASS(EXCEPTION HANDLING)\*\*\*\n";

  while (ch == 'y')

    {

      cout << " 1.Input . \n";

      cout << " 2.Multiplication of two matrices . \n";

      cout << " 3.Transpose of a matrix .\n";

      cout << " 4.Addition of two matrices . \n";

      cout << " 5.Subtraction of two matrices . \n\n";

      cout << " Enter your choice : ";

      cin >> x;

      switch (x)

    {

    case 1 :   o1.input ();

               o1.display ();

               o2.input ();

               o2.display ();

               o5.input ();

               o5.display ();

               break;

    case 2 : try

        {

         o3 = o1 \* o2;

         o3.display ();

        }

    catch (const char \*p)

         {

          cout << p;

         }

         break;

 case 3:

        try

        {

          o4 = o1.transpose ();

          cout << "Transpose of matrix is:- " << endl;

          o4.display ();

        }

    catch (const char \*p)

      {

        cout << p;

      }

        break;

case 4:

      try

      {

        o6 = o1 + o5;

        o6.display ();

      }

      catch (const char \*p)

      {

        cout << p;

      }

      break;

case 5:

      try

      {

        o6 = o1 - o5;

        o6.display ();

      }

      catch (const char \*p)

      {

        cout << p;

      }

      break;

    default:

     cout << " Doesn't find your choice,enter the correct choice ";

      break;}

    cout << "DO YOU WANT TO CONTINUE: ";

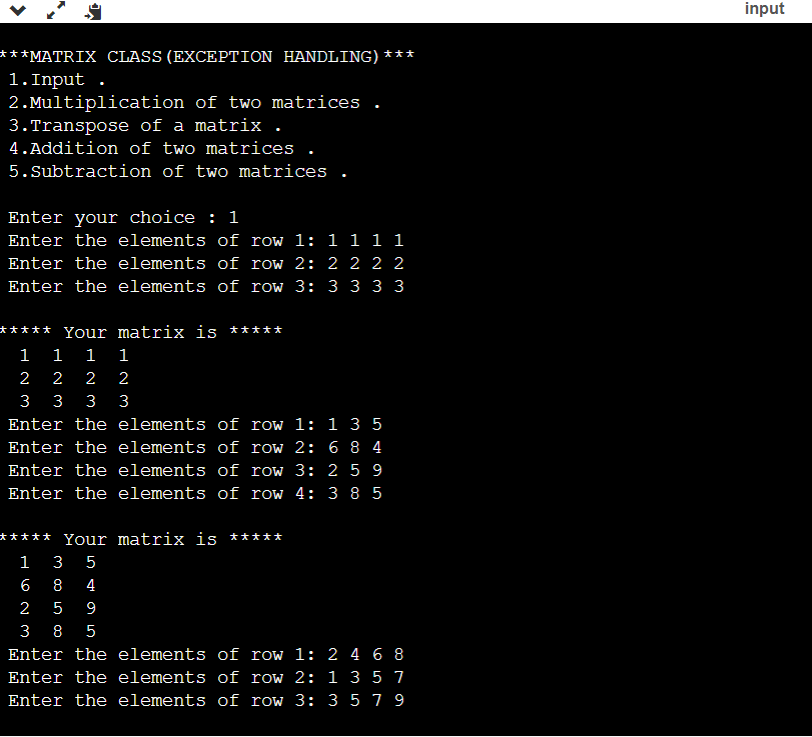
    cin >> ch;

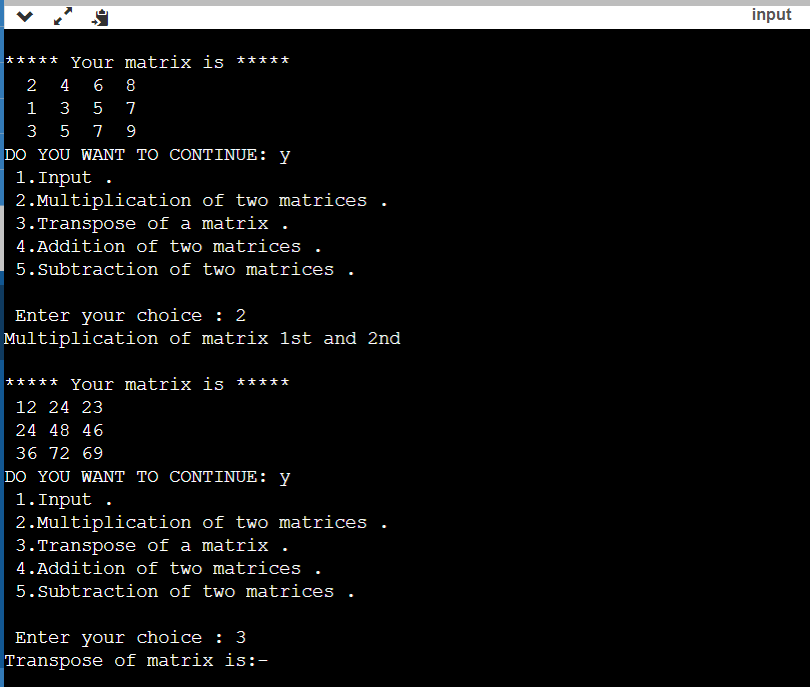
}

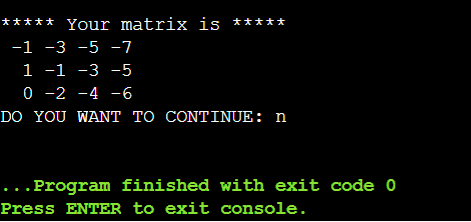
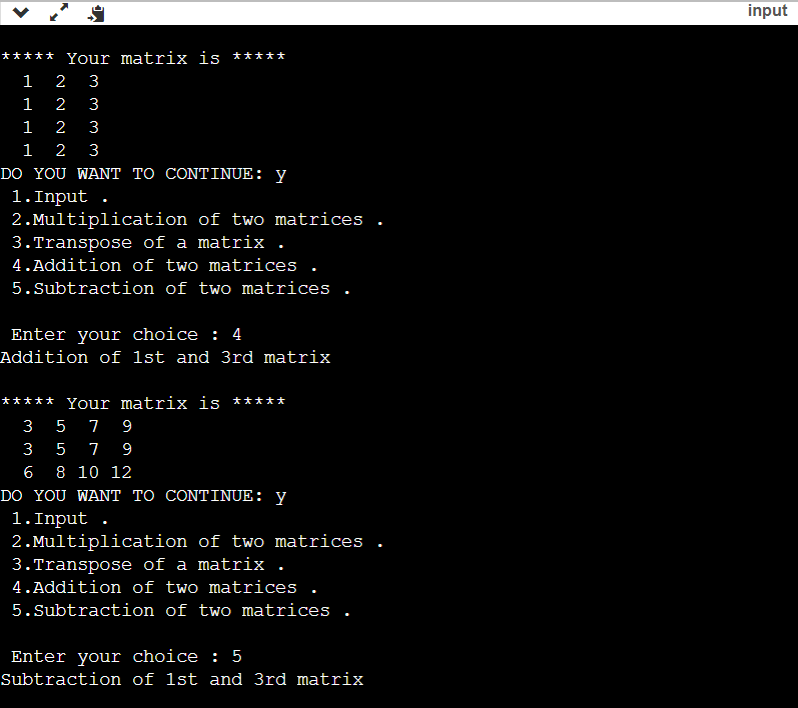
return 0;

}

**OUTPUT:-**







**QUES 13 Create a class student containing fields for roll number, name, class, year and total marks. Write a program to store five objects of student class in a file. Retrieve these records from file and display them.**

**=>**#include<iostream>

#include<iomanip>

#include<fstream>

using namespace std;

class student

{

   public:

   int Rollno,Year;

   char Name[20];

   char Class[20];

   double TotalMarks;

   void getinpt()

   {

     cout<<"Enter student details:- "<<endl;

     cout<<"\nEnter student roll number: ";

   cin>>Rollno;

   cout<<"Enter student year: ";

   cin>>Year;

   cout<<"Enter student name: ";

   cin>>Name;

   cout<<"Enter student class: ";

   cin>>Class;

   cout<<"Enter student total marks: ";

   cin>>TotalMarks;

   }

};

int main()

{

student s1[5];

   char ch;

   for(int i=0;i<3;i++)

   s1[i].getinpt();

   ofstream f1;

   f1.open("C:\\Users\\22sho\\OneDrive\\Desktop\\C++\\helloworld\\prog1\\program1\\Student.txt");

   if(!f1)

   {

     cout<<"Error in opening the read file!! ";

   exit(100);

   }

   for(int i=0;i<3;i++)

   {

     f1<<setw(5)<<s1[i].Rollno;

   f1<<setw(5)<<s1[i].Name;

   f1<<setw(5)<<s1[i].Class;

   f1<<setw(5)<<s1[i].Year;

   f1<<setw(5)<<s1[i].TotalMarks<<endl;

   }

   f1.close();

   ifstream f2;

   f2.open("C:\\Users\\22sho\\OneDrive\\Desktop\\C++\\helloworld\\prog1\\program1\\Student.txt");

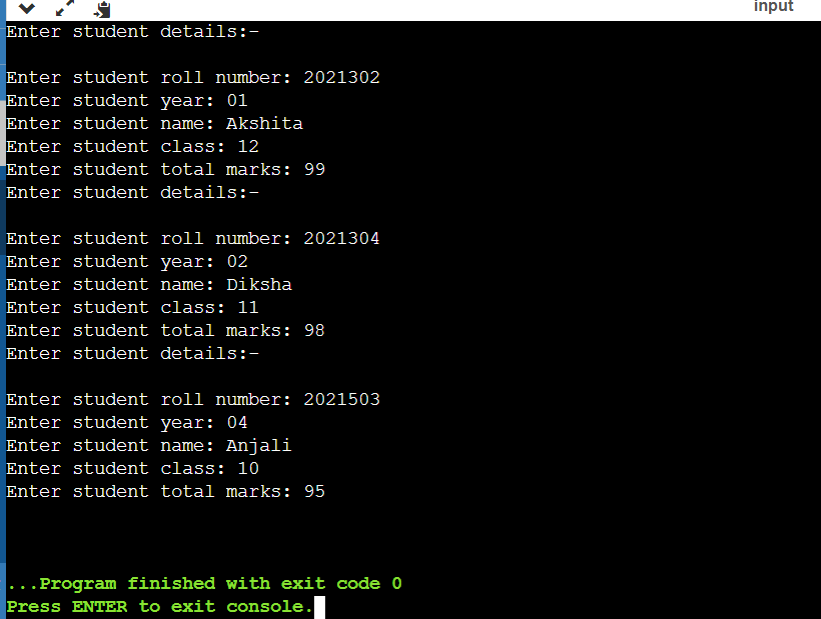
   while(f2.get(ch));

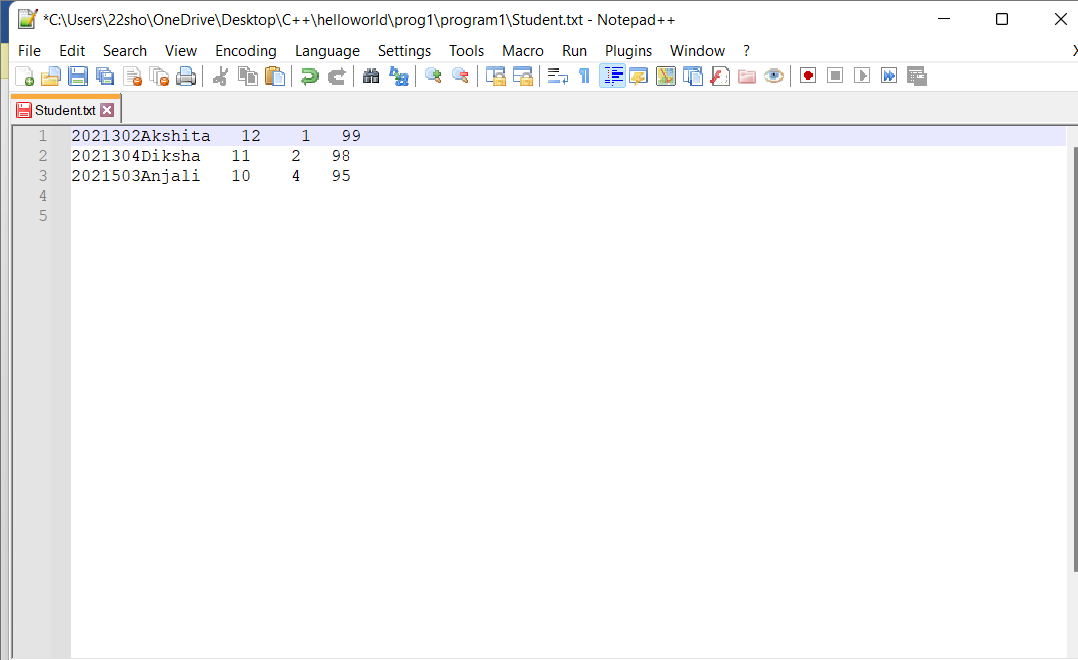
     cout<<ch;

   return 0;

}

**OUTPUT:-**





**QUES 14 - COPY THE CONTENTS OF ONE  TEXT FILE TO ANOTHER FILE, AFTER REMOVING ALL THE WHITESPACES.**

**=>**#include<iostream>

#include<fstream>

using namespace std;

int main()

{

  ifstream f1;

  ofstream f2;

  char ch;

  f1.open("C:\\Users\\22sho\\OneDrive\\Desktop\\C++\\helloworld\\prog1\\program1\\File 1.txt");

  if(!f1)

  { cout<<"Error in opening read file!!";

  exit(99);

  }

  f2.open("C:\\Users\\22sho\\OneDrive\\Desktop\\C++\\helloworld\\prog1\\program1\\File 2.txt")

  if(!f2)

  { cout<<"Error in opening a file!!";

  exit(100);

  }

  cout<<"write text\n";

  while(f1.get(ch))

  {

    if(ch!=' ' && ch!='\t' && ch!='\n' )

    f2.put(ch);

  }

  f1.close();

  f2.close();

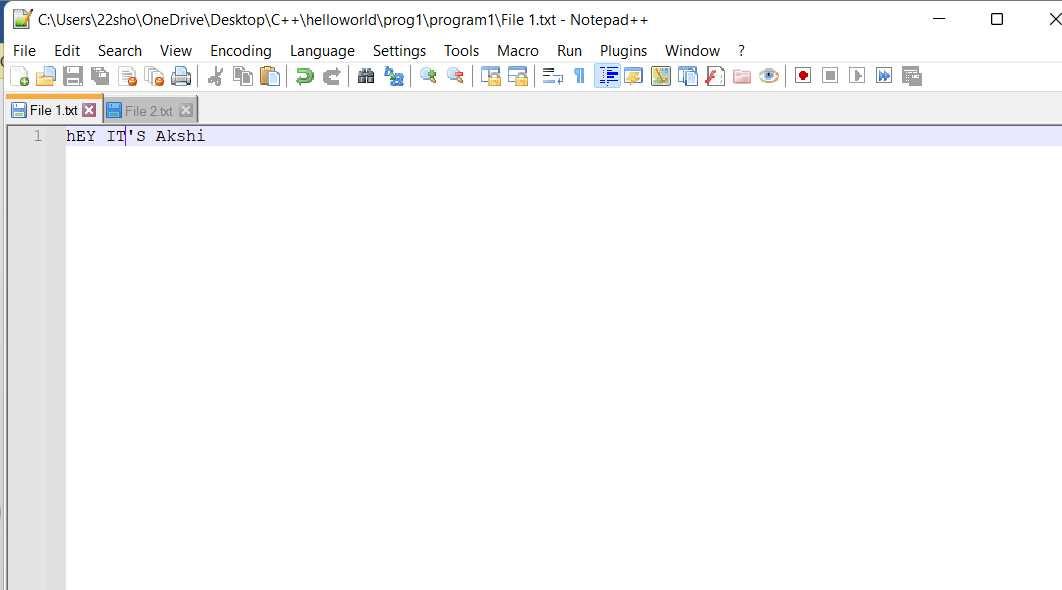
  cout<<"end";

   return 0;

}

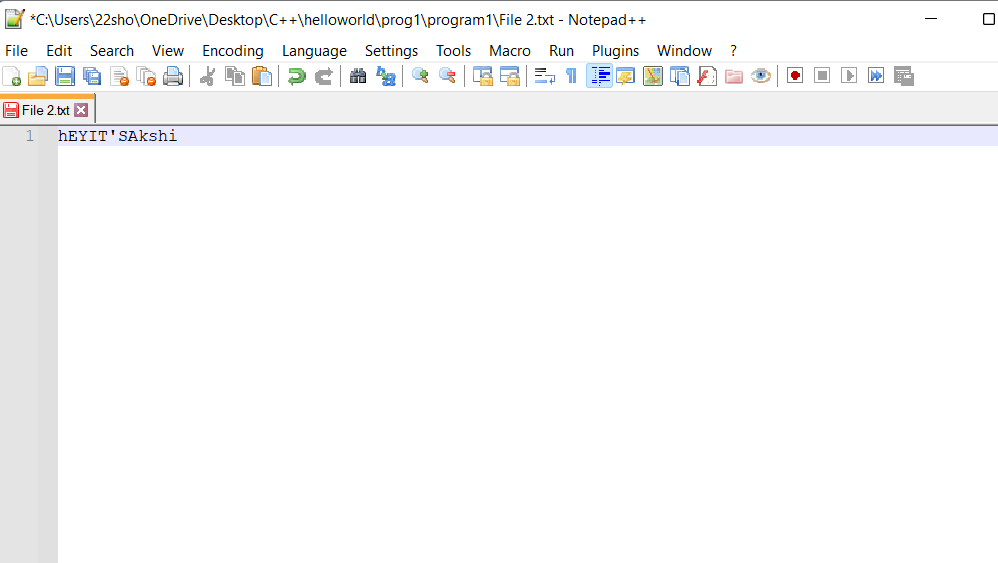
**OUTPUT:-**

**FILE 1.txt**

****

****

**File 2.txt**

****