

**Practical:1****1) Write programs to implement the following Substitution Cipher Techniques:****1a) Caesar Cipher**

Code:

```

package cn;

import java.util.Scanner;

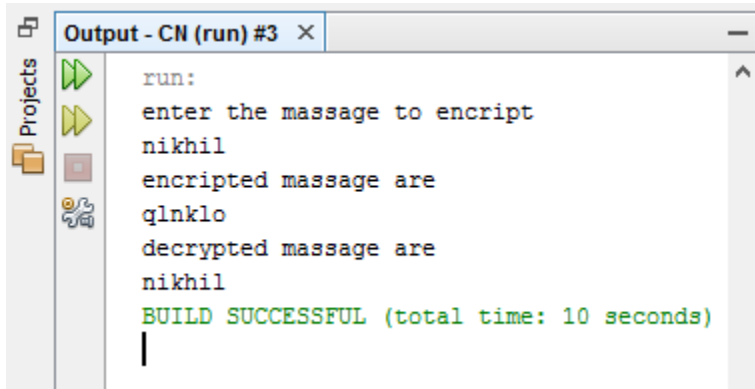
public class CN {

    public static final String Alphabet="abcdefghijklmnopqrstuvwxyz";

    public static String encrypt(String msg,int shiftValue)
    {
        String cipherText="";
        for(int i=0;i<msg.length();i++)
        {
            int pos=Alphabet.indexOf(msg.charAt(i));
            int cc=(pos+shiftValue)%26;
            char replace=Alphabet.charAt(cc);
            cipherText+=replace;
        }
        return cipherText;
    }
    public static String decrypt(String CipherText,int shiftKey)
    {
        String plainText="";
        for(int i=0;i<CipherText.length();i++)
        {
            int charPosition=Alphabet.indexOf(CipherText.charAt(i));
            int keyVal=(charPosition-shiftKey)%26;
            if(keyVal<0)
            {
                keyVal=Alphabet.length()+keyVal;
            }
            char replace=Alphabet.charAt(keyVal);
            plainText+=replace;
        }
        return plainText;
    }
    public static void main(String[] args) {
        // TODO code application logic here
        Scanner sc=new Scanner(System.in);
        System.out.println("enter the message to encrypt");
    }
}

```

```
String msg=new String();  
msg=sc.next();  
System.out.println("encrypted message are");  
System.out.println(encrypt(msg,3));  
System.out.println("decrypted message are");  
System.out.println(decrypt(encrypt(msg,3),3));  
  
}  
  
}
```

**Output:**

```
run:  
enter the message to encrypt  
nikhil  
encrypted message are  
qlnklo  
decrypted message are  
nikhil  
BUILD SUCCESSFUL (total time: 10 seconds)
```

## 1b) Monoalphabetic Cipher

Code:

```
package cn;

import static cn.CN.encrypt;
import java.util.Scanner;
public class monoalphabetic {
    public static char
plainAlpha[]={'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'};
    public static char
cipherAlpha[]={'z','y','x','w','v','u','t','s','r','q','p','o','n','m','l','k','j','i','h','g','f','e','d','c','b','a'};

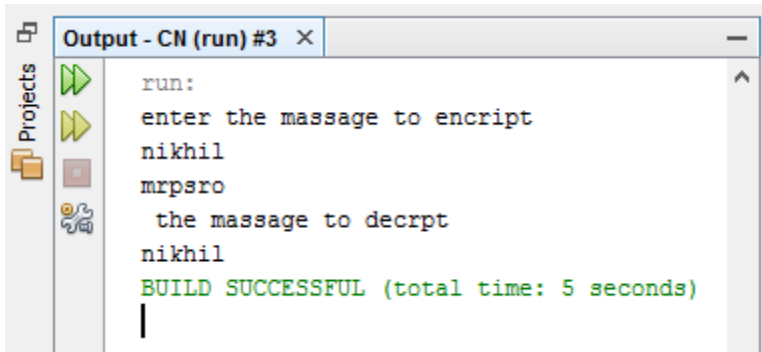
    public static String encrypt(String msg)
    {
        char cipherT[]=new char[(msg.length())];
        for(int i=0;i<msg.length();i++)
        {
            for(int j=0;j<26;j++)
            {
                if(plainAlpha[j]==msg.charAt(i))
                {
                    cipherT[i]=cipherAlpha[j];
                    break;
                }
            }
        }
        return (new String(cipherT));
    }
    public static String decrypt(String msg)
    {
        char plainT[]=new char[(msg.length())];
        for(int i=0;i<msg.length();i++)
        {
            for(int j=0;j<26;j++)
            {
                if(cipherAlpha[j]==msg.charAt(i))
                {
                    plainT[i]=plainAlpha[j];
                    break;
                }
            }
        }
        return (new String(plainT));
    }

    public static void main(String[] args) {
```

```
// TODO code application logic here
Scanner sc=new Scanner(System.in);
System.out.println("enter the message to encrypt");
String msg=new String();
String a=new String();
msg=sc.next();
System.out.println(encrypt(msg));
a=encrypt(msg);
System.out.println(" the message to decrpt");
System.out.println(decrypt(a));
sc.close();

}

}
```

**Output:**

```
run:
enter the message to encrypt
nikhil
mrpsro
the message to decrpt
nikhil
BUILD SUCCESSFUL (total time: 5 seconds)
```

**Practical:2****Aim: Write programs to implement the following Substitution Cipher Techniques****A) Vernam Cipher****B) Playfair Cipher****2A) Vernam Cipher****Code:**

```

package cn;
import java.util.Scanner;

public class VernamCipher {

    public static void main(String arg[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("enter plaintext");
        String plaintext=sc.nextLine().toUpperCase();
        System.out.println("enter key length should be equal to pt");
        String key=sc.nextLine().toUpperCase();
        String a=encrypt(plaintext,key);
        System.out.println("encrypted text are:"+a);
        String b=decrypt(a,key);
        System.out.println("decryptedt text are:"+b);
    }

    static String encrypt(String plaintext,String Key)
    {
        String ciphertext="";

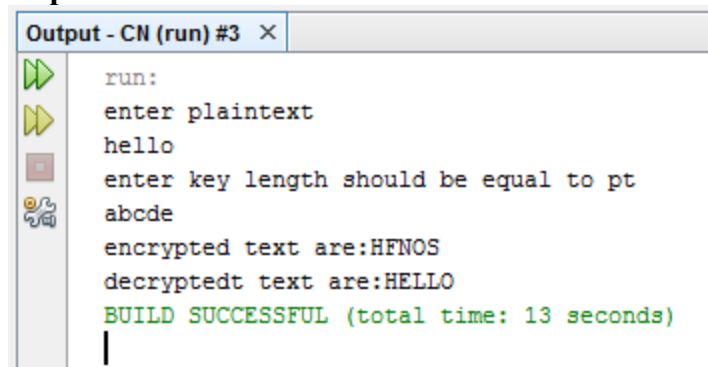
        int cipher[]=new int[Key.length()];
        for(int i=0;i<Key.length();i++)
        {
            cipher[i]=plaintext.charAt(i)-'A'+ Key.charAt(i)-'A';
        }
        for(int i=0;i<Key.length();i++)
        {
            if(cipher[i]>25)
            {
                cipher[i]=cipher[i]-26;
            }
        }
        for(int i=0;i<Key.length();i++)
    
```

```

    {
        int x=cipher[i]+'A';
        ciphertext+=(char)x;
    }
    return ciphertext;
}
static String decrypt(String ciphertext,String Key)
{
    String plaintext="";

    int plain[]=new int[Key.length()];
    for(int i=0;i<Key.length();i++)
    {
        plain[i]=(ciphertext.charAt(i)-'A') - (Key.charAt(i)-'A');
    }
    for(int i=0;i<Key.length();i++)
    {
        if(plain[i]<0)
        {
            plain[i]=plain[i]+26;
        }
    }
    for(int i=0;i<Key.length();i++)
    {
        int x=plain[i]+'A';
        plaintext+=(char)x;
    }
    return plaintext;
}
}

```

**Output:**


```

Output - CN (run) #3 x
run:
enter plaintext
hello
enter key length should be equal to pt
abcde
encrypted text are:HFNOS
decrypted text are:HELLO
BUILD SUCCESSFUL (total time: 13 seconds)
|

```

**2B) Playfair Cipher****Code:**

```

import java.awt.Point;
import java.util.Scanner;
public class PlayfairCipher
{
    private int length = 0;
    private String [][] table;
    public static void main(String args[])
    {
        PlayfairCipher pf = new PlayfairCipher();
    }
    private PlayfairCipher()
    {
        System.out.print("Enter the key for playfair cipher: ");
        Scanner sc = new Scanner(System.in);
        String key = parseString(sc);
        while(key.equals(""))
            key = parseString(sc);
        table = this.cipherTable(key);
        System.out.print("Enter the plaintext to be encipher: ");
        String input = parseString(sc);
        while(input.equals(""))
            input = parseString(sc);
        String output = cipher(input);
        String decodedOutput = decode(output);
        this.keyTable(table);
        this.printResults(output,decodedOutput);
    }
    private String parseString(Scanner sc)
    {
        {
            String parse = sc.nextLine();
            parse = parse.toUpperCase();
            parse = parse.replaceAll("[^A-Z]", "");
            parse = parse.replace("J", "I");
            return parse;
        }
    }
    private String[][] cipherTable(String key)
    {
        {
            String[][] playfairTable = new String[5][5];
            String keyString = key + "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
            for(int i = 0; i < 5; i++)
                for(int j = 0; j < 5; j++)
                    playfairTable[i][j] = "";
            for(int k = 0; k < keyString.length(); k++)
            {
                boolean repeat = false;

```

```

boolean used = false;
for(int i = 0; i < 5; i++)
{
    for(int j = 0; j < 5; j++)
    {
        if(playfairTable[i][j].equals("" + keyString.charAt(k)))
        {
            repeat = true;
        }
        else if(playfairTable[i][j].equals("") && !repeat && !used)
        {
            playfairTable[i][j] = "" + keyString.charAt(k);
            used = true;
        }
    }
}
return playfairTable;
}
private String cipher(String in)
{
    length = (int) in.length() / 2 + in.length() % 2;

    for(int i = 0; i < (length - 1); i++)
    {
        if(in.charAt(2 * i) == in.charAt(2 * i + 1))
        {
            in = new StringBuffer(in).insert(2 * i + 1, 'X').toString();
            length = (int) in.length() / 2 + in.length() % 2;
        }
    }
    String[] digraph = new String[length];
    for(int j = 0; j < length ; j++)
    {
        if(j == (length - 1) && in.length() / 2 == (length - 1))

        in = in + "X";
        digraph[j] = in.charAt(2 * j) + "" + in.charAt(2 * j + 1);
    }
    String out = "";
    String[] encDigraphs = new String[length];
    encDigraphs = encodeDigraph(digraph);
    for(int k = 0; k < length; k++)
    out = out + encDigraphs[k];
    return out;
}
private String[] encodeDigraph(String di[])

```



```

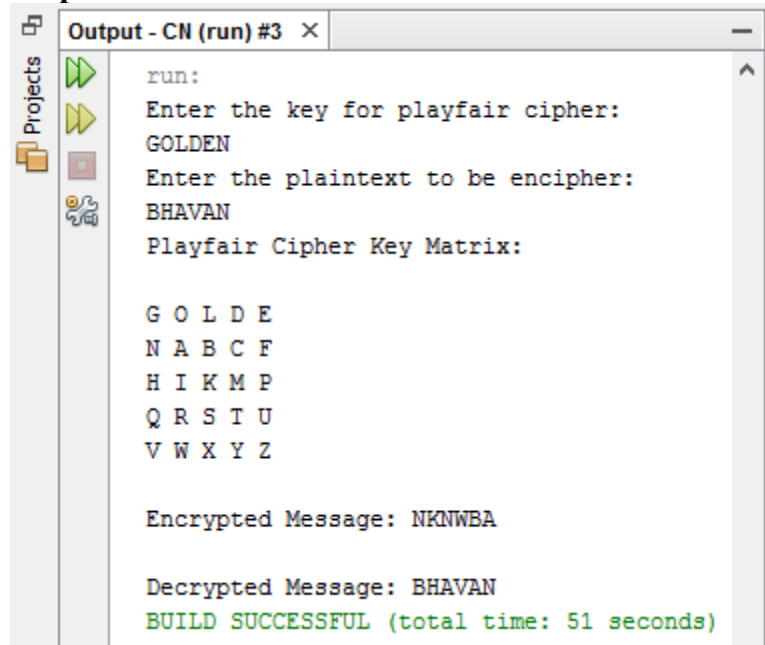
{
String[] encipher = new String[length];
for(int i = 0; i < length; i++)
{
char a = di[i].charAt(0);
char b = di[i].charAt(1);
int r1 = (int) getPoint(a).getX();
int r2 = (int) getPoint(b).getX();
int c1 = (int) getPoint(a).getY();
int c2 = (int) getPoint(b).getY();
if(r1 == r2)
{
c1 = (c1 + 1) % 5;
c2 = (c2 + 1) % 5;
}
else if(c1 == c2)
{
r1 = (r1 + 1) % 5;
r2 = (r2 + 1) % 5;
}
else
{
int temp = c1;
c1 = c2;
c2 = temp;
}
encipher[i] = table[r1][c1] + "" + table[r2][c2];
}
return encipher;
}
private String decode(String out)
{
String decoded = "";
for(int i = 0; i < out.length() / 2; i++)
{
char a = out.charAt(2*i);
char b = out.charAt(2*i+1);
int r1 = (int) getPoint(a).getX();
int r2 = (int) getPoint(b).getX();
int c1 = (int) getPoint(a).getY();
int c2 = (int) getPoint(b).getY();
if(r1 == r2)
{
c1 = (c1 + 4) % 5;
c2 = (c2 + 4) % 5;
}
else if(c1 == c2)
{

```

```

r1 = (r1 + 4) % 5;
r2 = (r2 + 4) % 5;
}
else
{
int temp = c1;
c1 = c2;
c2 = temp;
}
decoded = decoded + table[r1][c1] + table[r2][c2];
}
return decoded;
}
private Point getPoint(char c)
{
Point pt = new Point(0,0);
for(int i = 0; i < 5; i++)
for(int j = 0; j < 5; j++)
if(c == table[i][j].charAt(0))
pt = new Point(i,j);
return pt;
}
private void keyTable(String[][] printTable)
{
System.out.println("Playfair Cipher Key Matrix: ");
System.out.println();
for(int i = 0; i < 5; i++)
{
for(int j = 0; j < 5; j++)
{
System.out.print(printTable[i][j]+" ");
}
System.out.println();
}
System.out.println();
}
private void printResults(String encipher, String dec)
{
System.out.print("Encrypted Message: ");
System.out.println(encipher);
System.out.println();
System.out.print("Decrypted Message: ");
System.out.println(dec);
}
}

```

**Output:**

The screenshot shows an IDE's output window titled "Output - CN (run) #3". On the left, there is a "Projects" sidebar with icons for a project, a file, a class, and a test. The output text is as follows:

```
run:
Enter the key for playfair cipher:
GOLDEN
Enter the plaintext to be encipher:
BHAVAN
Playfair Cipher Key Matrix:

G O L D E
N A B C F
H I K M P
Q R S T U
V W X Y Z

Encrypted Message: NKNWBA

Decrypted Message: BHAVAN
BUILD SUCCESSFUL (total time: 51 seconds)
```

**Practical:3****Aim: Write programs to implement the following Transposition Cipher Techniques:****3a) Rail Fence Cipher****Code:**

```

package cn;
import java.util.Scanner;
public class RailFence
{
    public static void main(String ar[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the plain text: ");
        String plainText=sc.nextLine();

        System.out.println("Enter the key: ");
        int key=sc.nextInt();

        String text=encrypt(plainText,key);
        System.out.println(text);
        System.out.println(decrypt(text,key));
    }
    static String encrypt(String plainText,int key)
    {
        String cipherText="";
        boolean check=false;
        int j=0;
        int row=key;
        int col=plainText.length();
        char[][]a=new char[row][col];
        for(int i=0; i<col; i++)
        {
            if(j==0 || j==key-1)
                check=!check;
            a[j][i]=plainText.charAt(i);
            if(check)
                j++;
            else
                j--;
        }

        for(int i=0; i<row; i++)
        {
            for(int k=0; k<col; k++)
            {
                if(a[i][k]!=0)
                    cipherText+=a[i][k];
            }
        }
    }
}

```

```

    }

    for(int i=0; i<row; i++)
    {
        for(int k=0; k<col; k++)
        {
            System.out.print(a[i][k]);
        }
        System.out.println();
    }
    return cipherText;
}

static String decrypt(String cipherText,int key)
{
    String plainText="";
    boolean check=false;
    int j=0;
    int row=key;
    int col=cipherText.length();
    char[][]a=new char[row][col];
    for(int i=0; i<col; i++)
    {
        if(j==0 || j==key-1)
            check=!check;
        a[j][i]='*';
        if(check)
            j++;
        else
            j--;
    }
    for(int i=0; i<row; i++)
    {
        for(int k=0; k<col; k++)
        {
            System.out.print(a[i][k]);
        }
        System.out.println();
    }
    int index=0;
    check=false;
    for(int i=0; i<row; i++)
    {
        for(int k=0; k<col; k++)
        {
            if(a[i][k]=='*&&index<col)
            {
                a[i][k]=cipherText.charAt(index++);
            }
        }
    }
}

```

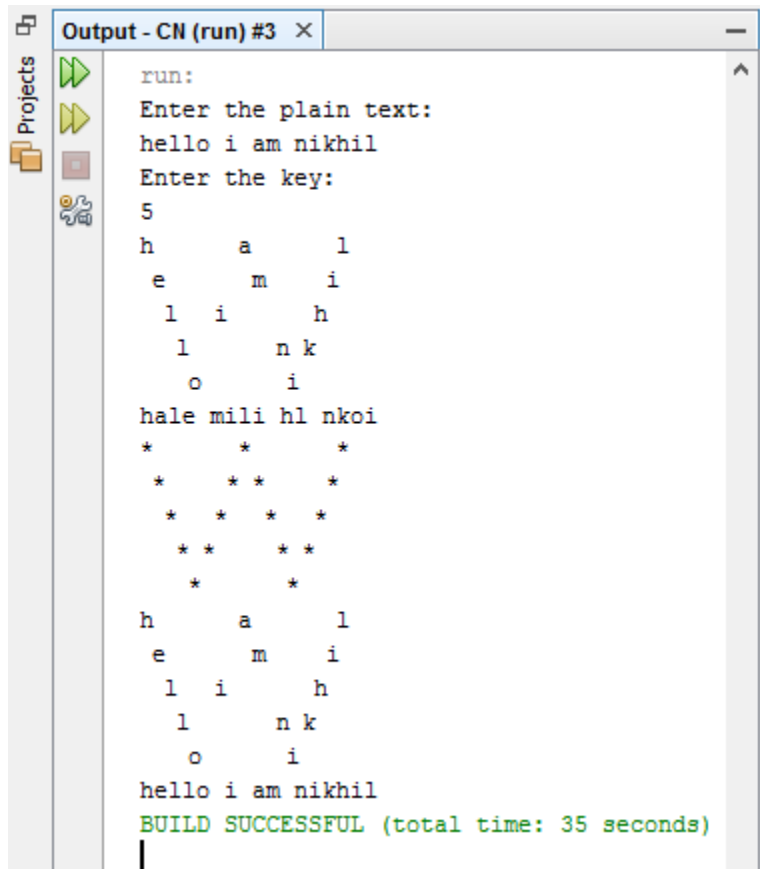
```

        }
    }
    for(int i=0; i<row; i++)
    {
        for(int k=0; k<col; k++)
        {
            System.out.print(a[i][k]);

        }
        System.out.println();
    }
    j=0;
    for(int i=0; i<col; i++)
    {
        if(j==0 || j==key-1)
            check=!check;
        plainText+=a[j][i];
        if(check)
            j++;
        else
            j--;
    }
    return plainText;
}
}

```

**Output:**



The screenshot shows a C++ IDE with a project named 'CN'. The output window displays the execution of a program that performs a Vigenere cipher encryption. The user enters the plain text 'hello i am nikhil' and a key '5'. The program then displays the plain text in a spaced-out format, followed by a decorative border of asterisks, and finally the encrypted text 'hale mili hl nkoi'. The program concludes with a 'BUILD SUCCESSFUL' message indicating a total execution time of 35 seconds.

```
run:
Enter the plain text:
hello i am nikhil
Enter the key:
5
h      a      l
e      m      i
l  i      h
l      n k
o      i
hale mili hl nkoi
*      *      *
*      *      *
*      *      *
*      *      *
*      *      *
h      a      l
e      m      i
l  i      h
l      n k
o      i
hello i am nikhil
BUILD SUCCESSFUL (total time: 35 seconds)
```

**3b) Simple Columnar Technique :**

Code:

```

package cn;

import java.util.Scanner;
public class SimpleCol {

    public static void main(String ar[])
    {
        Scanner sc= new Scanner(System.in);
        System.out.println("Enter plain text: ");
        String plaintext= sc.nextLine();
        System.out.println("Enter keyword: ");
        String key= sc.nextLine();
        encrypt(plaintext,key);
        String c=encrypt(plaintext,key);
        decrypt(c,key);
    }
    static String getKeycolIndex(String key, int keyInteger[])
    {
        String index="";

        for(int i=1;i<key.length()+1;i++)
        {
            for(int j=0;j<key.length();j++)
            {
                if(keyInteger[j]==i)
                    index+=j;
            }
        }
        return index;
    }

    static int[] keyInteger(String key)
    {
        String alphabets= "abcdefghijklmnopqrstuvwxyz";
        int []keyIntegerList= new int[key.length()];
        int r=0;
        for(int i=0;i<alphabets.length();i++)
        {
            for(int j=0;j<key.length();j++)
            {
                if(alphabets.charAt(i)==key.charAt(j))
                {
                    r++;
                    keyIntegerList[j]= r;
                }
            }
        }
    }
}

```



```

        }
    }
}
return keyIntegerList;
}

static String encrypt(String plaintext, String key)
{
    String ciphertext="";
    int k=0;
    int col= key.length();
    int row= plaintext.length();
    char a[][]= new char[row][col];

    for(int i=0;i<row;i++)
    {
        for(int j=0;j<col;j++)
        {
            if(k<plaintext.length())
            {
                a[i][j]= plaintext.charAt(k);
                k++;
            }
            else
                break;
        }
    }
    for(int i=0;i<row;i++)
    {
        for(int j=0;j<col;j++)
        {
            System.out.print(a[i][j]);
        }
        System.out.println();
    }
    int[] keyIntegerList= keyInteger(key);
    for(int c:keyIntegerList)
    {
        System.out.print(c);
    }

    String index= getkeycolIndex(key,keyIntegerList);
    System.out.println("\n");
    System.out.println(index);

    for(int s=0,w=0;s<row;s++,w++)
    {
        int d;

```

```

        if(w==key.length())
            break;
        else
            d=Character.getNumericValue(index.charAt(w));

        for(int j=0;j<row;j++)
        {
            ciphertext+=a[j][d];
        }
    }
    System.out.println("Cipher Text: "+ciphertext);
    return ciphertext;
}

static void decrypt(String ciphertext, String key)
{
    int row=ciphertext.length()/key.length();

    int col=key.length();
    char ciphertextM[][]=new char[row][col];

    int[] keyIntegerList = keyInteger(key);

    String index = getKeyColIndex(key, keyIntegerList);

    for (int i = 0, k = 0; i < ciphertext.length(); i++, k++)
    {
        int d = 0;
        if (k == key.length())
        {
            k = 0;
        } else
        {
            d = Character.getNumericValue(index.charAt(k));
        }

        for (int j = 0; j < row; j++, i++)
        {
            ciphertextM[j][d] = ciphertext.charAt(i);
        }
        --i;
    }

    String plainText = "";

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

```

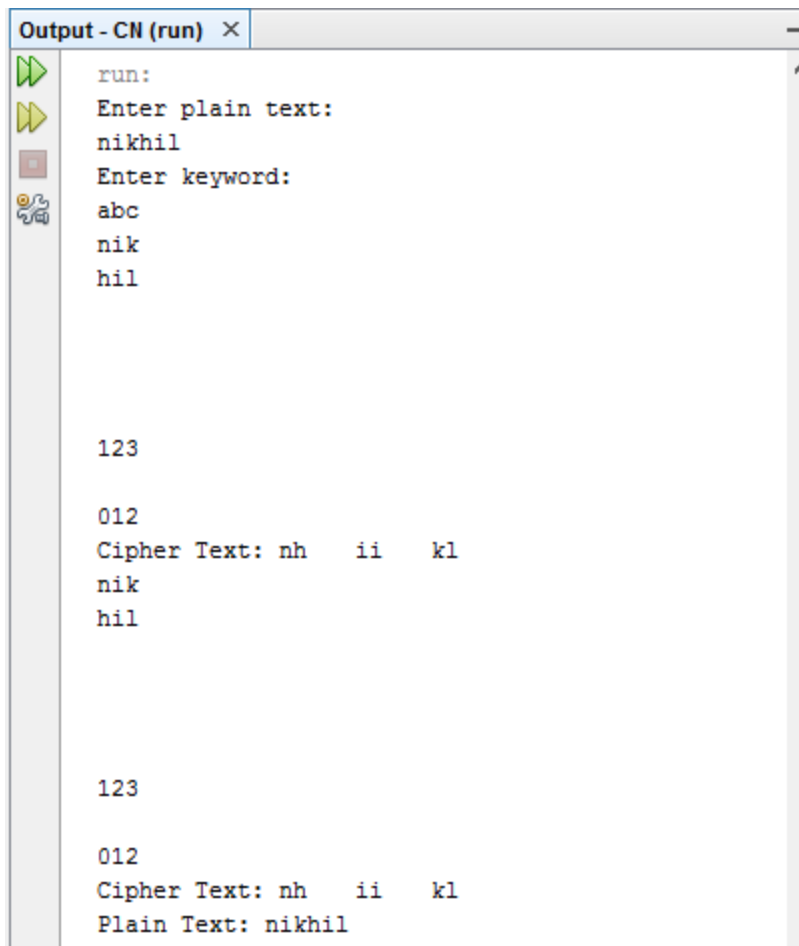
```

        {
        for (int j = 0; j < key.length(); j++)
            {
            plainText+=ciphertextM[i][j];
            }
        }
    }

    System.out.println("Plain Text: " + plainText);

}
}

```

**Output:**


```

run:
Enter plain text:
nikhil
Enter keyword:
abc
nik
hil

123

012
Cipher Text: nh    ii    kl
nik
hil

123

012
Cipher Text: nh    ii    kl
Plain Text: nikhil

```

**Practical:4**

**Aim:** Write program to encrypt and decrypt strings using

- a) DES Algorithm
- b) AES Algorithm

**a) DES**

**code:**

```
package cn;

import java.util.*;
import java.security.*;
import javax.crypto.*;
import javax.crypto.spec.*;

public class DES
{
    byte[] skey=new byte[1000];
    String skeyString;
    static byte[] raw;

    public DES()
    {
        try{
            Scanner sc=new Scanner(System.in);
            System.out.println("Enter plain text: ");
            String plaintext=sc.nextLine();
            byte[] pt=plaintext.getBytes();
            generateKey();

            byte[] ciphertext=encrypt(raw,pt);
            String encryptedData = new String(ciphertext);
            System.out.println("Encrypted message "+encryptedData);

            byte[] dbyte= decrypt(raw,ciphertext);
            String decryptedMessage = new String(dbyte);
            System.out.println("Decrypted message "+decryptedMessage);
        }
        catch(Exception e){System.out.println(e);}
    }

    void generateKey()throws Exception
    {
        Random r=new Random();
        int num=r.nextInt(10000);
    }
}
```

```

        String snum=String.valueOf(num);
        byte[] bnum=snum.getBytes();
        skey=getRawKey(bnum);
    }

    static byte[] getRawKey(byte[] seed)throws Exception
    {
        KeyGenerator kgen=KeyGenerator.getInstance("DES");
        SecureRandom sr=SecureRandom.getInstance("SHA1PRNG");
        sr.setSeed(seed);
        kgen.init(56,sr);
        SecretKey skey=kgen.generateKey();
        raw=skey.getEncoded();
        return raw;
    }

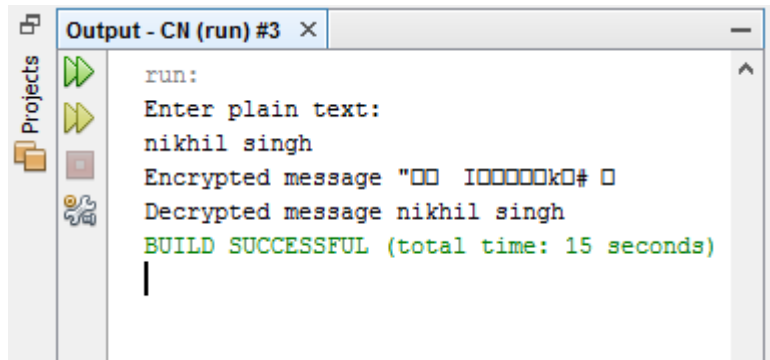
    private static byte[] encrypt(byte[] raw,byte[] msg)throws Exception
    {
        SecretKeySpec keyspec=new SecretKeySpec(raw,"DES");
        Cipher c=Cipher.getInstance("DES");
        c.init(Cipher.ENCRYPT_MODE,keyspec);
        byte[] encrypted=c.doFinal(msg);
        return encrypted;
    }

    private static byte[] decrypt(byte[] raw,byte[] encrypted)throws Exception
    {
        SecretKeySpec keyspec=new SecretKeySpec(raw,"DES");
        Cipher c=Cipher.getInstance("DES");
        c.init(Cipher.DECRYPT_MODE,keyspec);
        byte[] decrypted=c.doFinal(encrypted);
        return decrypted;
    }

    public static void main(String arg[])
    {
        DES obj=new DES();
    }
}

```

**output:**



The screenshot shows an IDE's output window titled "Output - CN (run) #3". On the left, a "Projects" sidebar contains icons for a project folder, a run button (green play), a debug button (green play with bug), a stop button (red square), and a refresh button (circular arrow). The output text is as follows:

```
run:
Enter plain text:
nikhil singh
Encrypted message "  I00000k0# 
Decrypted message nikhil singh
BUILD SUCCESSFUL (total time: 15 seconds)
```

**b)AES Algorithm****code:**

```

package cn;
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Base64;

import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.util.*;

public class AES {

    private static SecretKeySpec secretKey;
    private static byte[] key;

    public static void setKey(String myKey)
    {
        MessageDigest sha = null;
        try {
            key = myKey.getBytes("UTF-8");
            sha = MessageDigest.getInstance("SHA-1");
            key = sha.digest(key);
            key = Arrays.copyOf(key, 16);
            secretKey = new SecretKeySpec(key, "AES");
        }
        catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }

    public static String encrypt(String strToEncrypt, String secret)
    {
        try
        {
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES");
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);

```

```

        return Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-
8"))));
    }
    catch (Exception e)
    {
        System.out.println("Error while encrypting: " + e.toString());
    }
    return null;
}

public static String decrypt(String strToDecrypt, String secret)
{
    try
    {
        setKey(secret);
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
        cipher.init(Cipher.DECRYPT_MODE, secretKey);
        return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
    }
    catch (Exception e)
    {
        System.out.println("Error while decrypting: " + e.toString());
    }
    return null;
}

    public static void main(String[] args)
{
    final String secretKey = "ssshhhhhhhhhhh!!!!";

    Scanner sc=new Scanner(System.in);
        System.out.println("Enter plain text: ");
        String originalString=sc.nextLine();

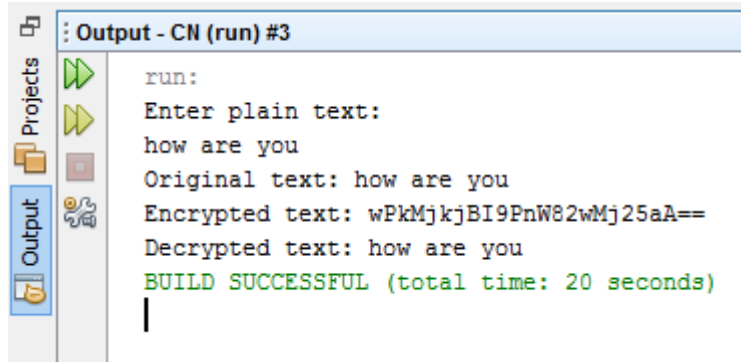
        String encryptedString = AES.encrypt(originalString, secretKey) ;
        String decryptedString = AES.decrypt(encryptedString, secretKey) ;

        System.out.println("Original text: "+originalString);
        System.out.println("Encrypted text: "+encryptedString);
        System.out.println("Decrypted text: "+decryptedString);
    }
}

```

**output:**





```
run:
Enter plain text:
how are you
Original text: how are you
Encrypted text: wPkMjkjBI9PnW82wMj25aA==
Decrypted text: how are you
BUILD SUCCESSFUL (total time: 20 seconds)
|
```

**Practical:5**

**Aim:** Write a program to implement RSA algorithm to perform encryption / decryption of a given string.

**Code:**

```
package cn;

import java.math.BigInteger;
import java.util.Random;
import java.io.*;

public class RSA {

    private BigInteger p;
    private BigInteger q;
    private BigInteger N;
    private BigInteger phi;
    private BigInteger e;
    private BigInteger d;
    private int bitlength = 1024;

    private Random r;
    public RSA() {
        r = new Random();
        p = BigInteger.probablePrime(bitlength, r);
        q = BigInteger.probablePrime(bitlength, r);
        N = p.multiply(q);

        phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
        e = BigInteger.probablePrime(bitlength/2, r);

        while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0 ) {
            e.add(BigInteger.ONE);
        }
        d = e.modInverse(phi);
    }

    public static void main (String[] args) throws IOException
    {
        RSA rsa = new RSA();
        DataInputStream in=new DataInputStream(System.in);
        String teststring ;
        System.out.println("Enter the plain text:");
        teststring=in.readLine();
        System.out.println("Encrypting String: " + teststring);
        System.out.println("String in Bytes: " + bytesToString(teststring.getBytes()));

        // encrypt
```

## INFORMATION AND NETWORK SECURITY PRACTICALS

```
byte[] encrypted = rsa.encrypt(teststring.getBytes());
System.out.println("Encrypted String in Bytes: " + bytesToString(encrypted));

// decrypt
byte[] decrypted = rsa.decrypt(encrypted);
System.out.println("Decrypted String in Bytes: " + bytesToString(decrypted));

System.out.println("Decrypted String: " + new String(decrypted));

}

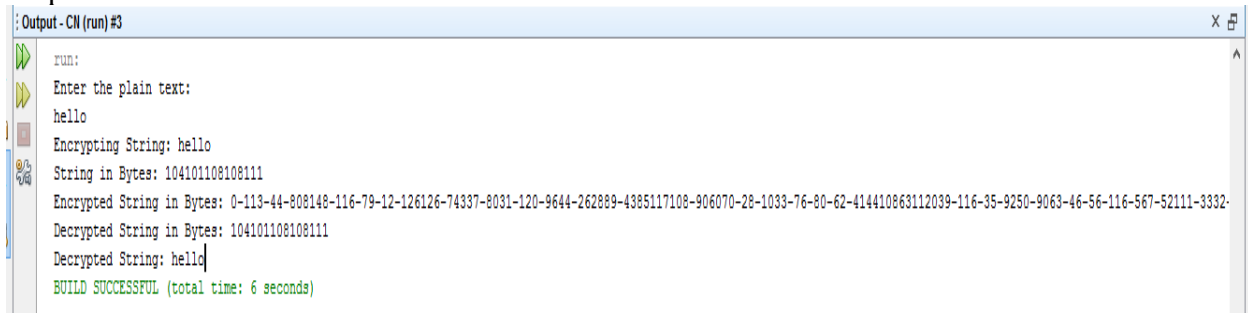
private static String bytesToString(byte[] encrypted) {
    String test = "";
    for (byte b : encrypted) {
        test += Byte.toString(b);
    }
    return test;
}

public byte[] encrypt(byte[] message) {
    return (new BigInteger(message)).modPow(e, N).toByteArray();
}

public byte[] decrypt(byte[] message) {
    return (new BigInteger(message)).modPow(d, N).toByteArray();
}

}
```

### Output:



```
Output - CN (run) #3
run:
Enter the plain text:
hello
Encrypting String: hello
String in Bytes: 104101108108111
Encrypted String in Bytes: 0-113-44-808148-116-79-12-126126-74337-8031-120-9644-262889-4385117108-906070-28-1033-76-80-62-414410863112039-116-35-9250-9063-46-56-116-567-52111-3332-
Decrypted String in Bytes: 104101108108111
Decrypted String: hello
BUILD SUCCESSFUL (total time: 6 seconds)
```

## Practical:6

**Aim:** Write a program to implement the Diffie-Hellman Key Agreement algorithm to generate symmetric keys

**Code:**

```
package cn;

import java.util.Scanner;
public class diffie {
    public static void main(String arg[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("enter prime number for n");
        int n = sc.nextInt();

        System.out.println("enter prime number for g");
        int g = sc.nextInt();

        System.out.println("enter value for x");
        int x = sc.nextInt();

        System.out.println("enter value for y");
        int y = sc.nextInt();

        int A=(int)Math.pow(g,x)%n;
        System.out.println("value of A:"+A);

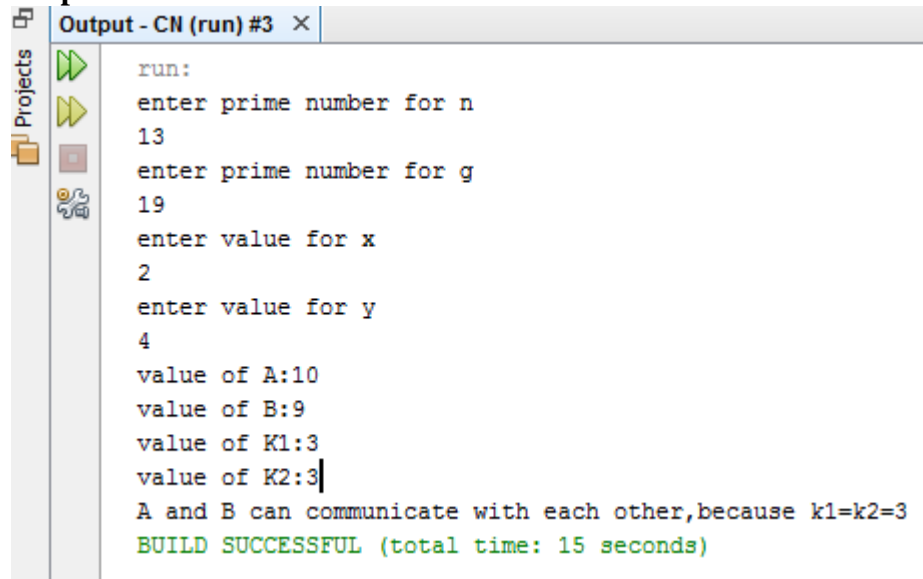
        int B=(int)Math.pow(g,y)%n;
        System.out.println("value of B:"+B);

        int K1=(int)Math.pow(B,x)%n;
        System.out.println("value of K1:"+K1);

        int K2=(int)Math.pow(A,y)%n;
        System.out.println("value of K2:"+K2);

        if(K1==K2)
        {
            System.out.println("A and B can communicate with each other,because k1=k2="+K1);
        }
        else{
            System.out.println("A and B not communicate with each other");
        }
    }
}
```

```
}
```

**Output:**

The screenshot shows an IDE's output window titled "Output - CN (run) #3". On the left, a vertical toolbar contains icons for running (a green play button), stepping through (a yellow play button), stopping (a red square), and debugging (a magnifying glass over a bug). The output text is as follows:

```
run:
enter prime number for n
13
enter prime number for g
19
enter value for x
2
enter value for y
4
value of A:10
value of B:9
value of K1:3
value of K2:3
A and B can communicate with each other,because k1=k2=3
BUILD SUCCESSFUL (total time: 15 seconds)
```

## Practical:7

**Aim:** Write a program to implement the MD5 algorithm compute the message digest

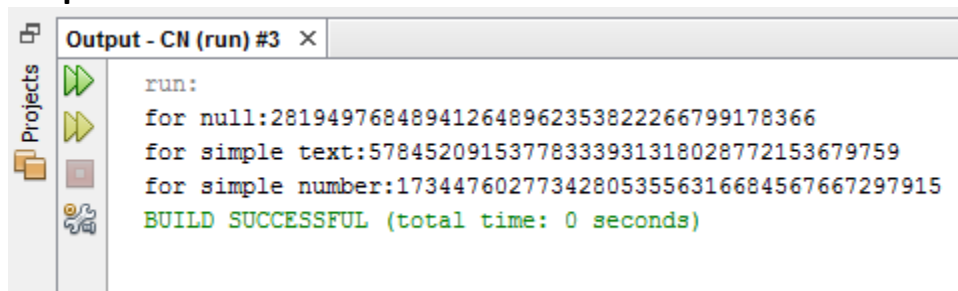
### Code:

```
package cn;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.logging.Level;
import java.util.logging.Logger;

public class MD5 {
    public static void main(String arg[])
    {
        System.out.println("for null:"+ md5(""));
        System.out.println("for simple text:"+ md5("good morning"));
        System.out.println("for simple number:"+ md5("12345"));
    }
    public static String md5(String input)
    {
        String md5=null;
        if(null==input) return null;
        MessageDigest md;
        try {
            md = MessageDigest.getInstance("MD5");
            md.update(input.getBytes(),0,input.length());
            md5=new BigInteger(1,md.digest()).toString();

        } catch (NoSuchAlgorithmException ex) {
            Logger.getLogger(MD5.class.getName()).log(Level.SEVERE, null, ex);
        }
        return md5;
    }
}
```

### Output:



```
run:
for null:281949768489412648962353822266799178366
for simple text:57845209153778333931318028772153679759
for simple number:173447602773428053556316684567667297915
BUILD SUCCESSFUL (total time: 0 seconds)
```

## Practical:8

**Aim:** Write a program to calculate HMAC-SHA1 Signature

**Code:**

```
package cn;
import java.security.*;
import java.util.Formatter;
import javax.crypto.*;
import javax.crypto.spec.*;

public class HmacShalSignature {

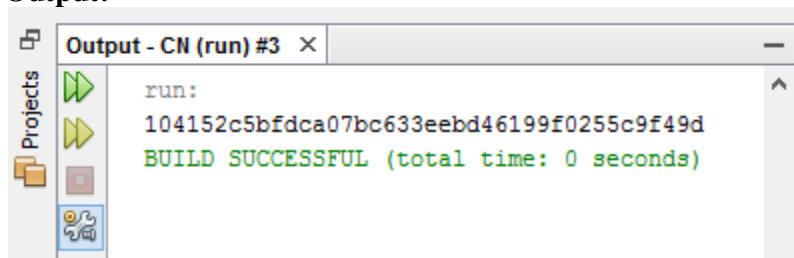
    private static String toHexString(byte[] bytes)
    {
        Formatter formatter =new Formatter();

        for(byte b:bytes)
        {
            formatter.format("%02x", b);
        }
        return formatter.toString();
    }
    public static String calculateHMACCode(String data,String key)
        throws SignatureException,NoSuchAlgorithmException,InvalidKeyException
    {
        SecretKeySpec signingKey=new SecretKeySpec(key.getBytes(),"HmacSHA1");
        Mac mac=Mac.getInstance("HmacSHA1");
        mac.init(signingKey);

        return toHexString(mac.doFinal(data.getBytes()));
    }
    public static void main(String[] args) throws Exception{
        String hmac=calculateHMACCode("data","key");

        System.out.println(hmac);
    }
}
```

**Output:**



**Practical:9**

**Aim:** Write a program to implement SSL

**Server:**

```
package cn;

import java.io.BufferedReader;
import java.io.BufferedInputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {
    private Socket soc;
    private ServerSocket ss;
    private DataInputStream dis;

    public Server(int port)
    {
        try{
            ss=new ServerSocket(port);
            System.out.println("server started.");
            System.out.println("waiting for client.");
            soc=ss.accept();
            System.out.println("client accepted");
            dis=new DataInputStream(new BufferedInputStream(soc.getInputStream()));

            String msg="";
            while(!msg.equals("End"))
            {
                msg=dis.readUTF();
                System.out.println(msg);
            }
            soc.close();
            dis.close();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }

    public static void main(String arg[])
    {
        Server obj=new Server(2000);
    }
}
```



```

    }
}

```

### Client:

```

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.Socket;

public class Client {

    private Socket soc;
    private DataOutputStream dos;
    private DataInputStream dis;

    public Client(String ipaddr,int port)
    {
        try{
            soc=new Socket(ipaddr,port);
            System.out.println("connected");
            dis=new DataInputStream(System.in);
            dos=new DataOutputStream(soc.getOutputStream());

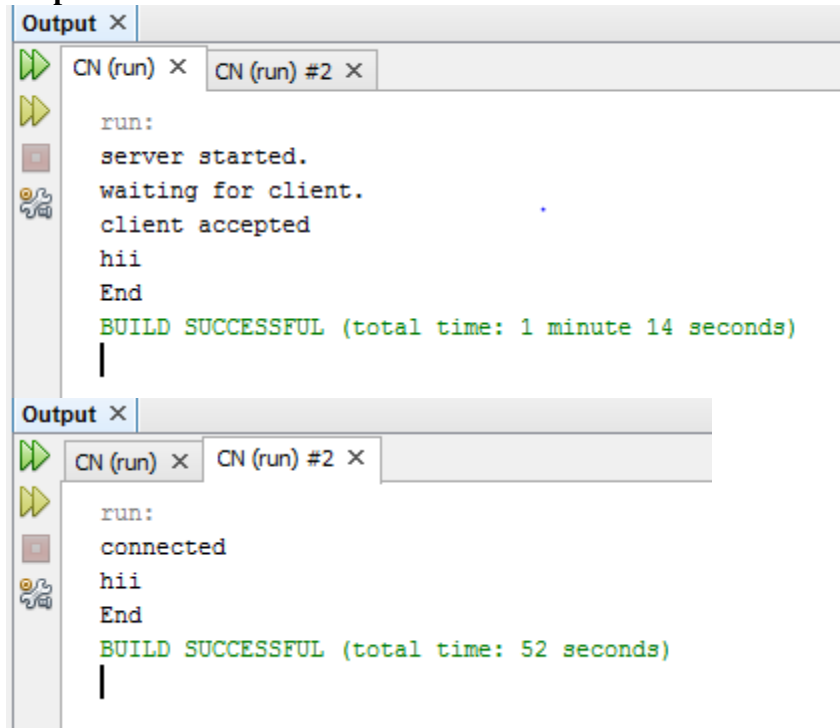
            String msg="";
            while(!msg.equals("End"))
            {
                msg=dis.readLine();
                dos.writeUTF(msg);

            }
            soc.close();
            dis.close();
            dos.close();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
    public static void main(String ar[])
    {
        Client ob=new Client("localhost",2000);
    }
}

```

```
}
```

```
}
```

**Output:**

```
Output X
CN (run) X CN (run) #2 X
run:
server started.
waiting for client.
client accepted
hii
End
BUILD SUCCESSFUL (total time: 1 minute 14 seconds)
|

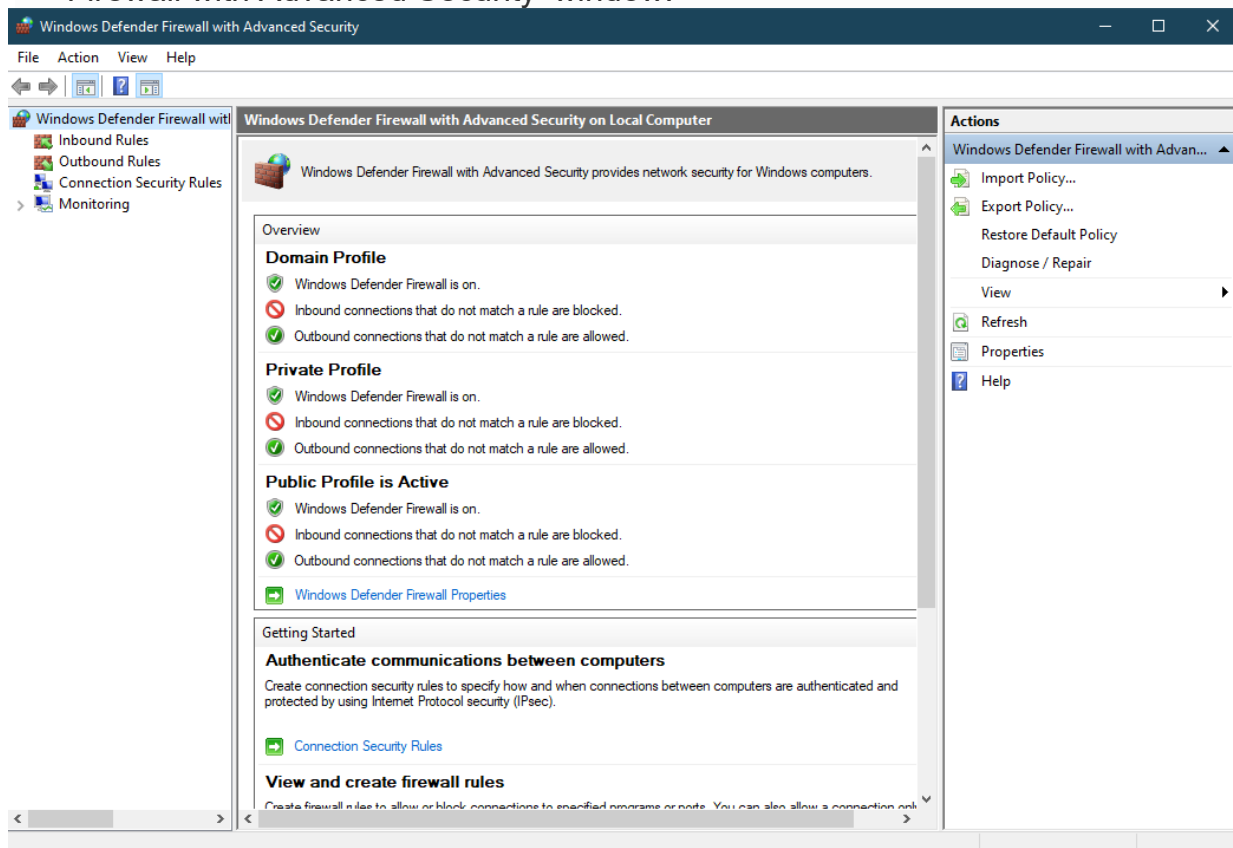
Output X
CN (run) X CN (run) #2 X
run:
connected
hii
End
BUILD SUCCESSFUL (total time: 52 seconds)
|
```

**Practical:10****Aim: Configure Windows Firewall to block:****A )port-**

- 1. Open Windows Firewall and find the Advanced Settings.**

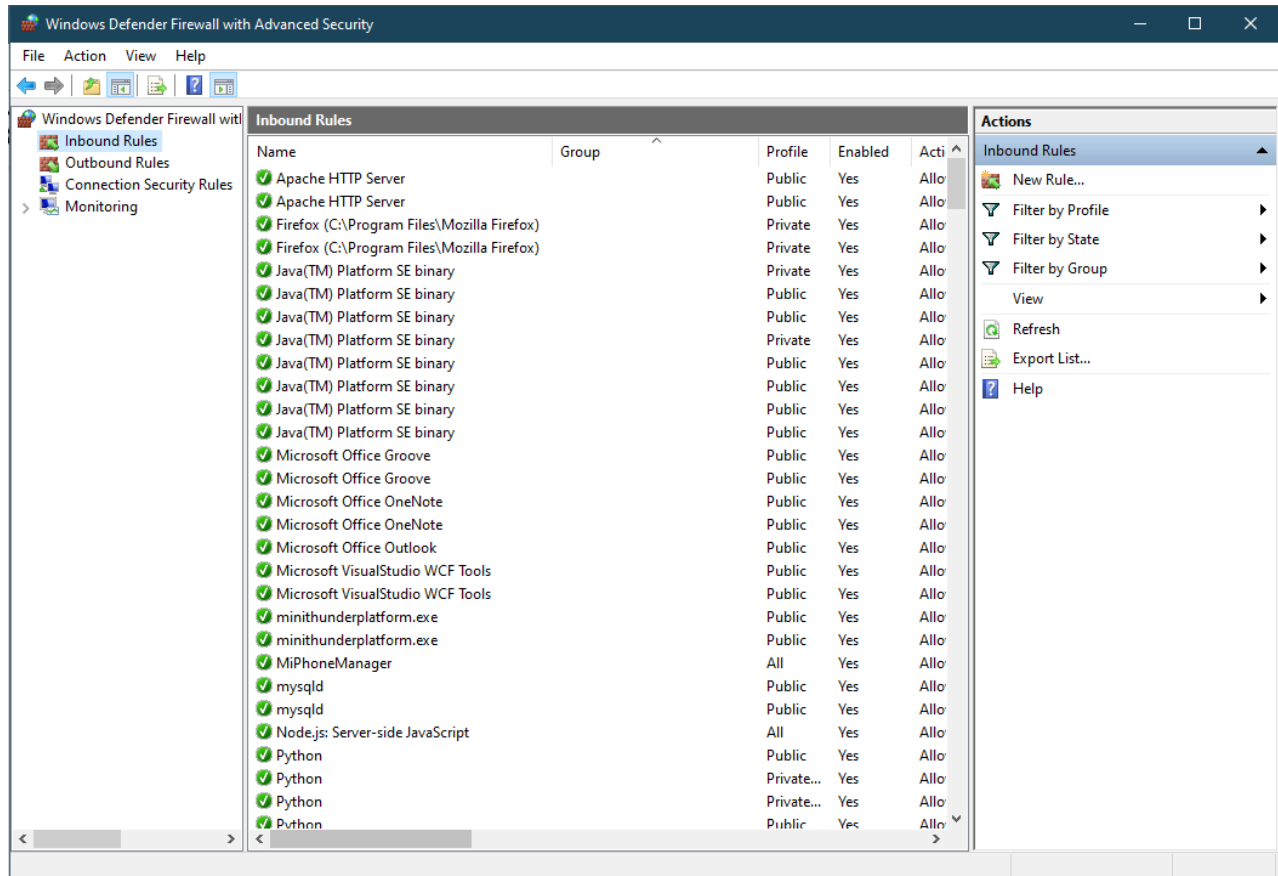
To open Windows Firewall, type 'firewall.cpl' into the search bar and press the Enter key.

When 'Advanced Settings' opens, click the **Advanced Settings** link in the left-hand pane of the main firewall dialog box. This will bring up the 'Windows Firewall with Advanced Security' window.



## 2. Open the List of Inbound Rules.

On the left-hand pane of the window, click on 'Inbound Rules' to bring up the list of rules.



### 3. Set up a New Rule.

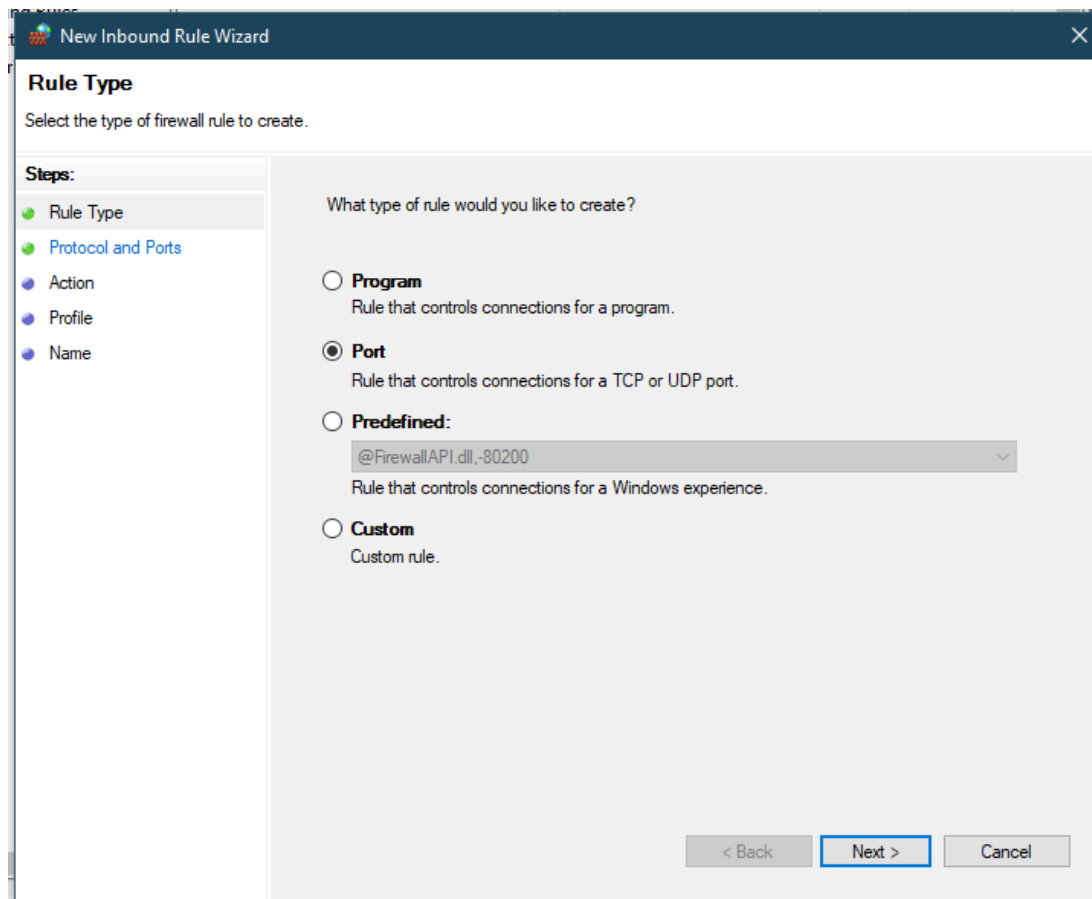
From the Actions pane on the right-hand side, select 'New Rule...'

Select 'Port' and then click 'Next.' This will open the 'New Inbound Rule Wizard' window.

From there, select 'Port' as the new Rule Type and click 'Next.'

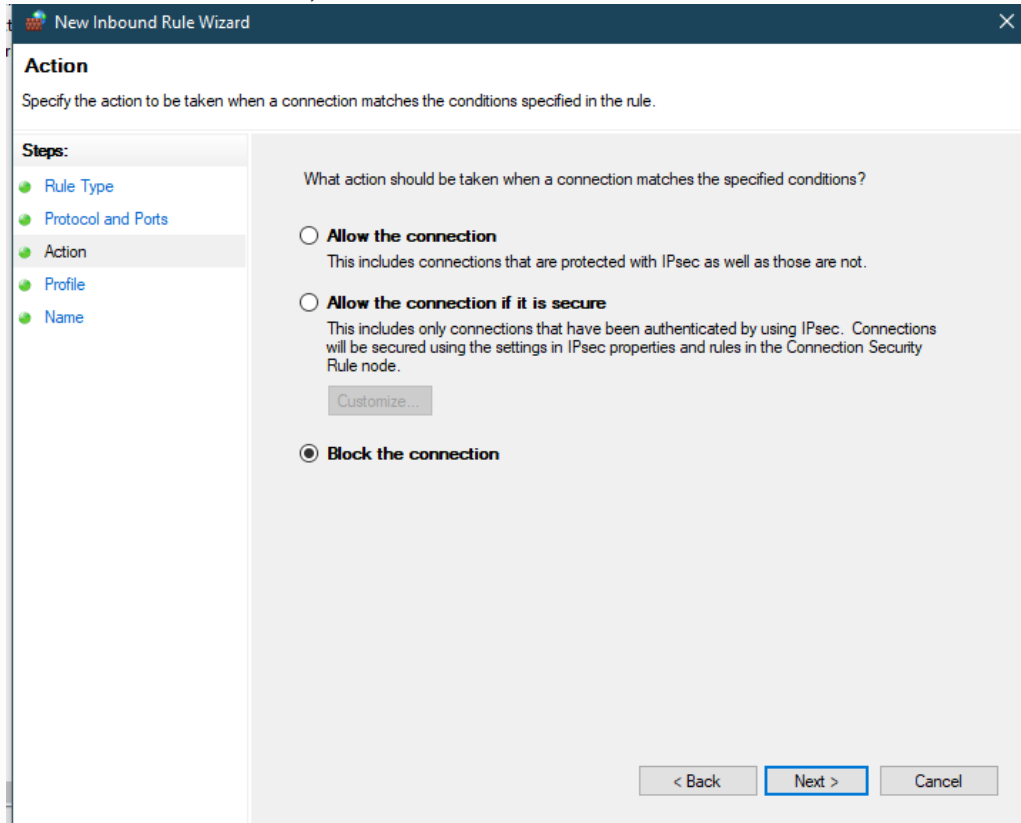
Click on 'Specific local ports.' Then choose a port number (e.g., 80).

Click 'Next' to continue.



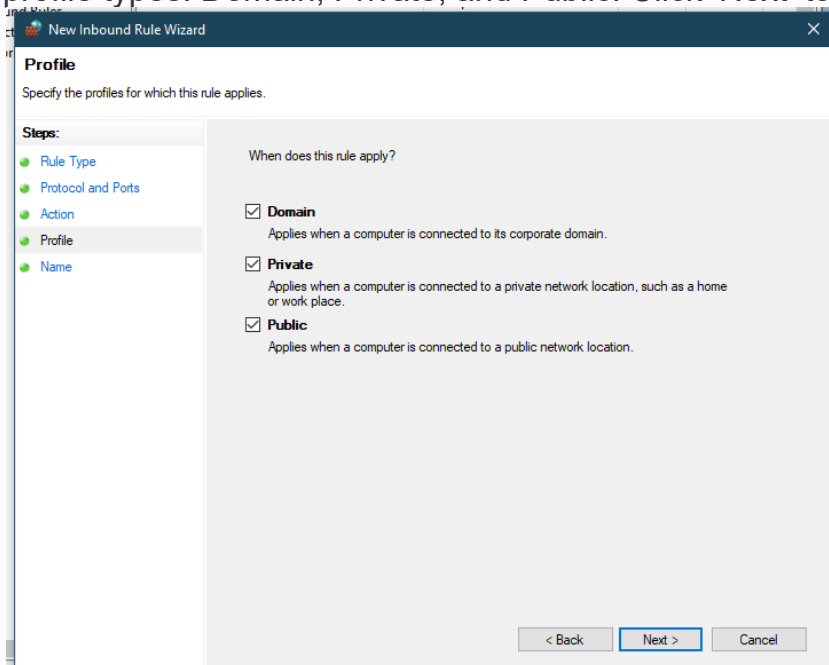
#### 4. Block the Connection.

In the Action window, select 'Block the connection' and click 'Next.'



#### 5. Apply Your New Rule to Each Profile Type.

In the Profile window, tick the boxes to apply your rule to each of the three profile types: Domain, Private, and Public. Click 'Next' to continue.



## 6. Name Your Rule and Configure the Settings.

Choose a name for your new rule, e.g., 'block suspicious ports.'  
If you want, you can also add an optional description to your rule.  
When you're done, click 'Finish' to configure the settings.

New Inbound Rule Wizard

**Name**

Specify the name and description of this rule.

**Steps:**

- Rule Type
- Protocol and Ports
- Action
- Profile
- Name**

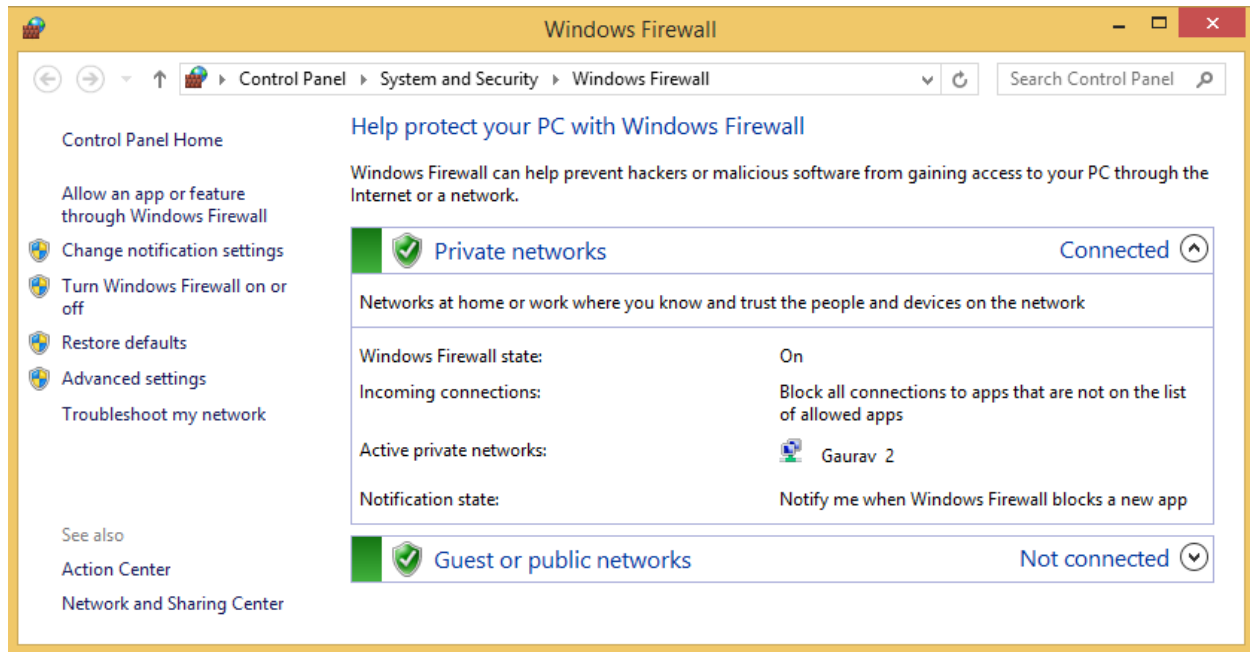
Name:  
suspicious ports

Description (optional):

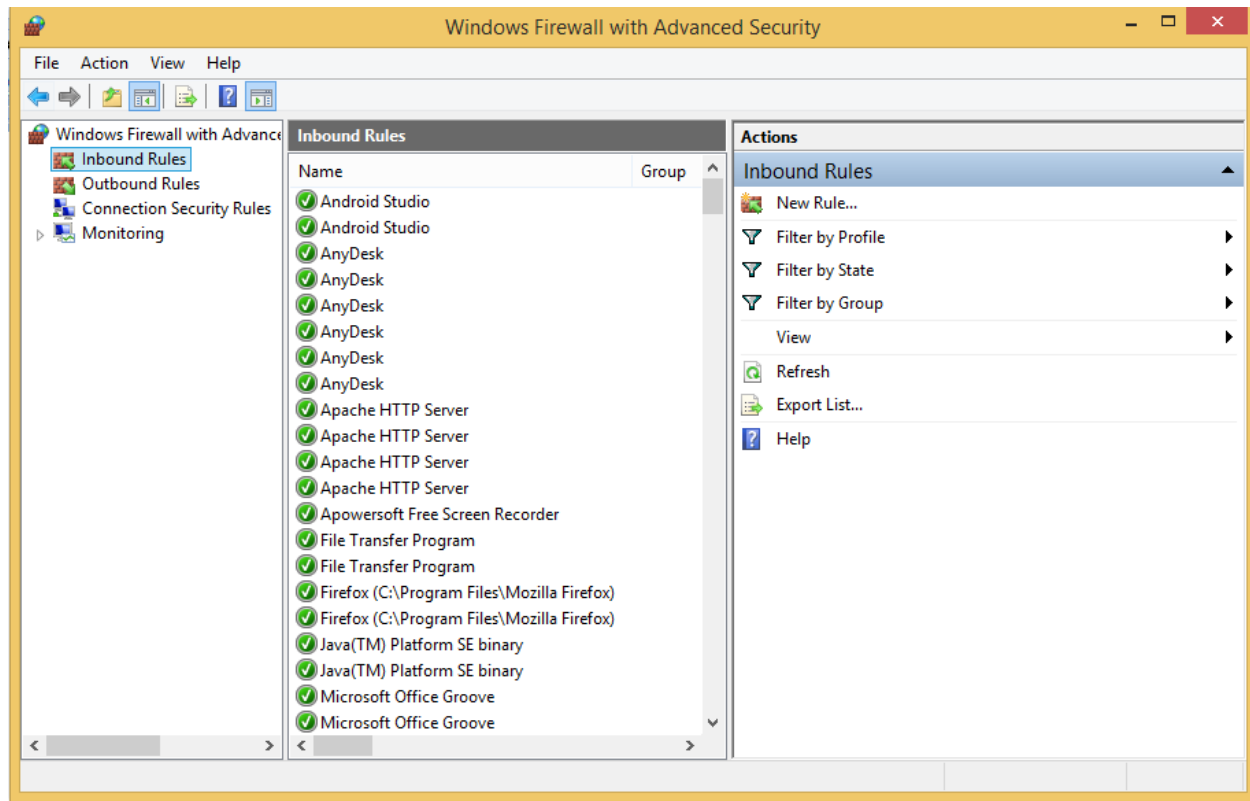
< Back   Finish   Cancel

## B) -A Program

Step 1: Open Windows Defender Firewall and then click on Advance settings.

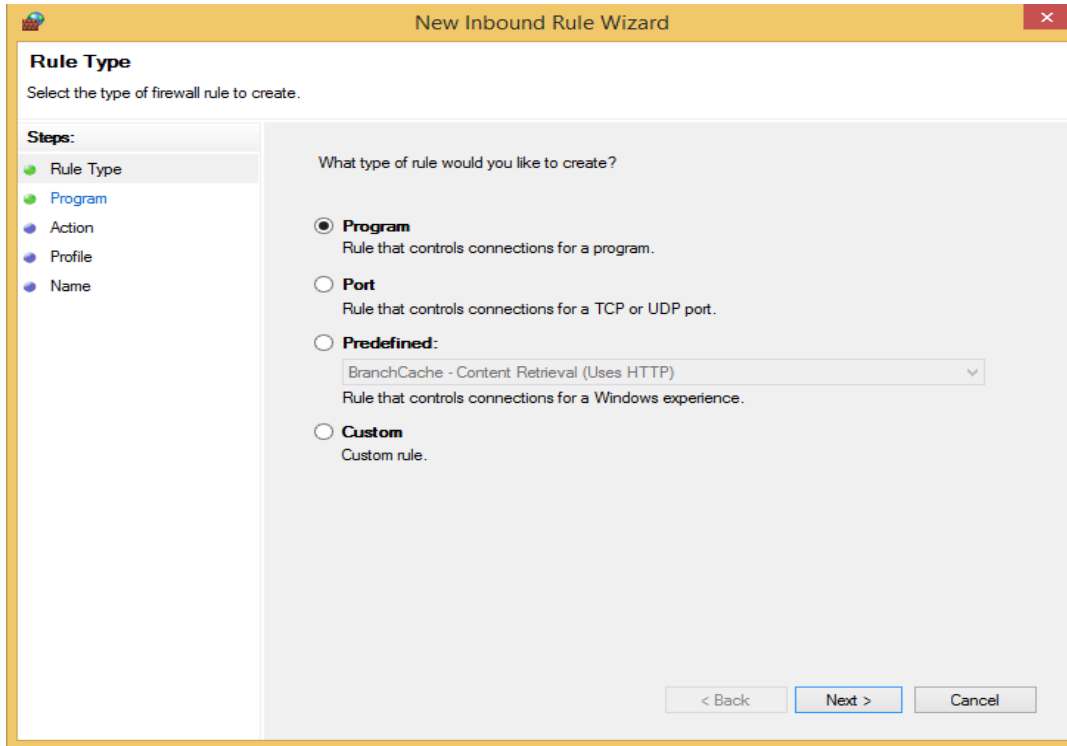


2) Click on Inbound rule and then click on New rule.

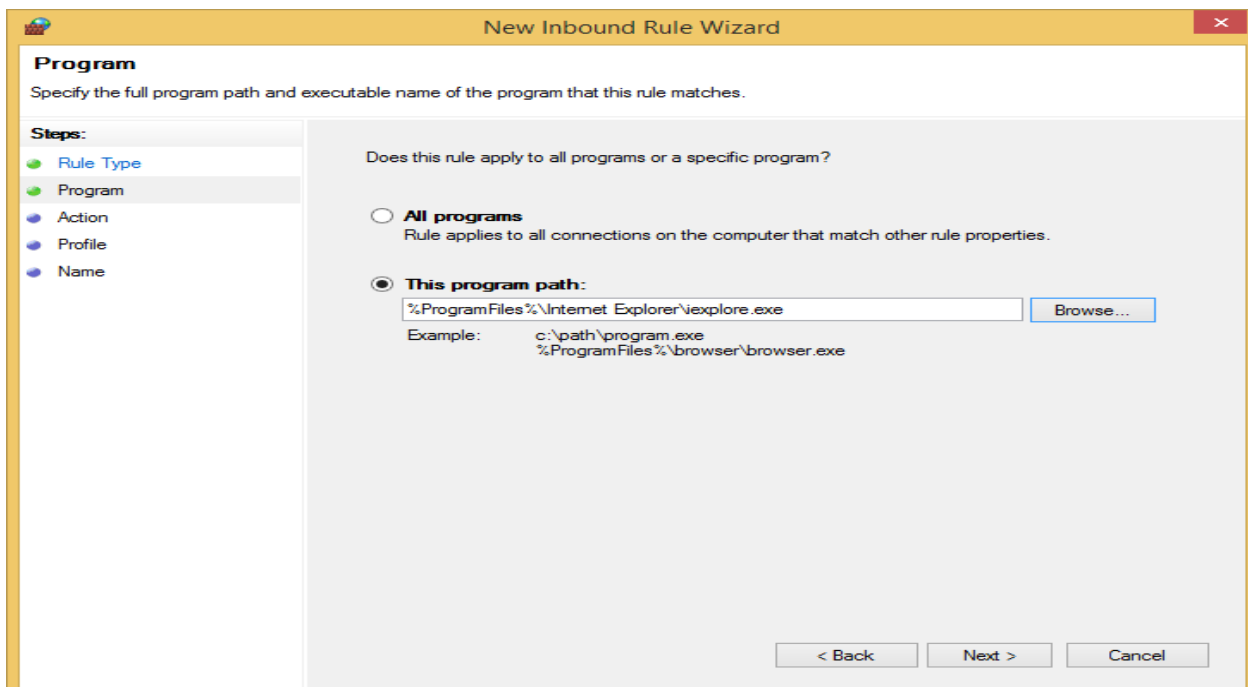




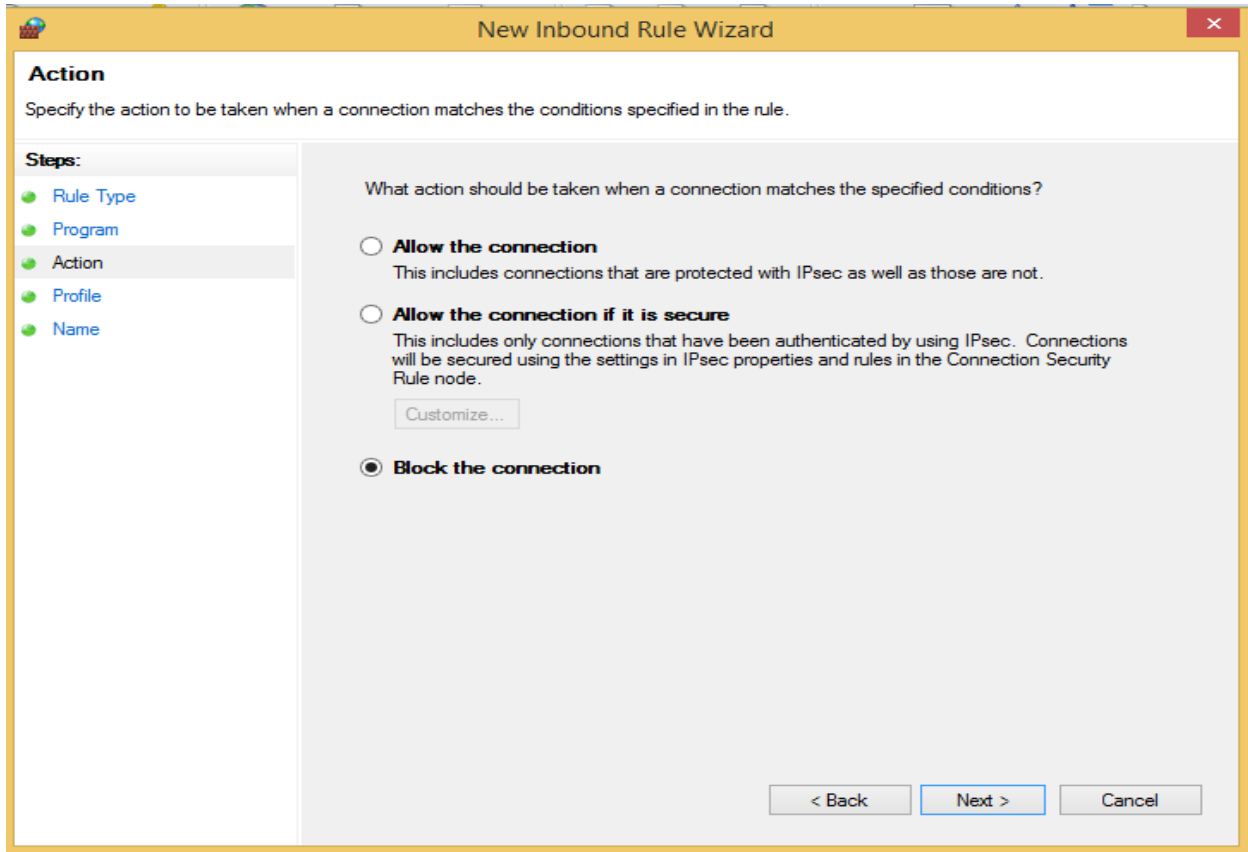
3) Select Program and then click on next.



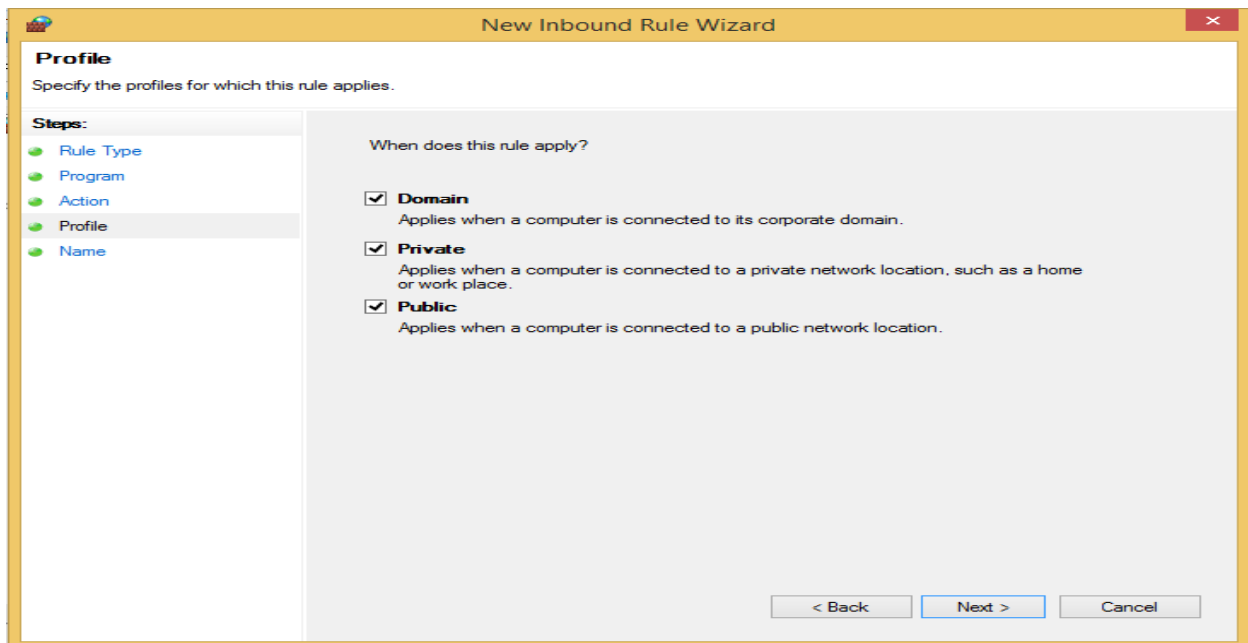
4) Select “This program path” and then click on “Browse” and then select the browser’s .exe file and then click on next.



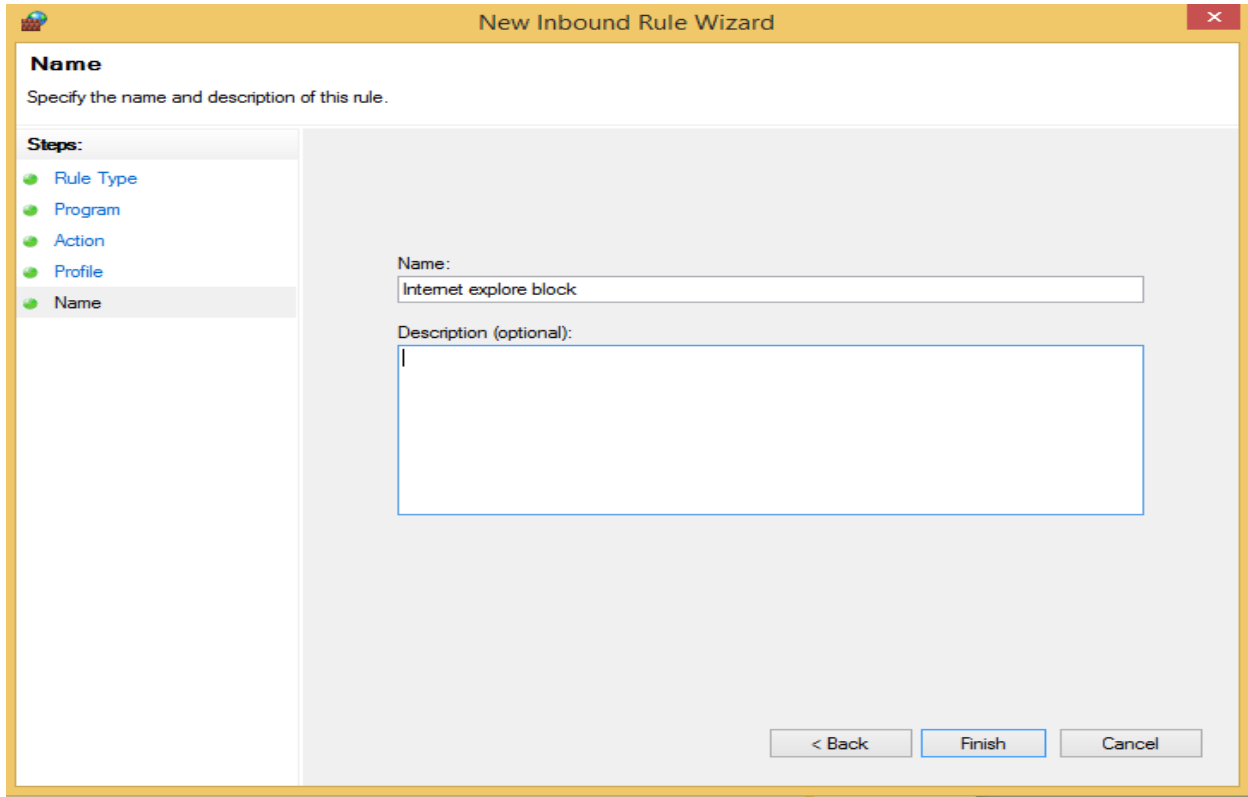
5) Select “Block the connection” and then click on next.



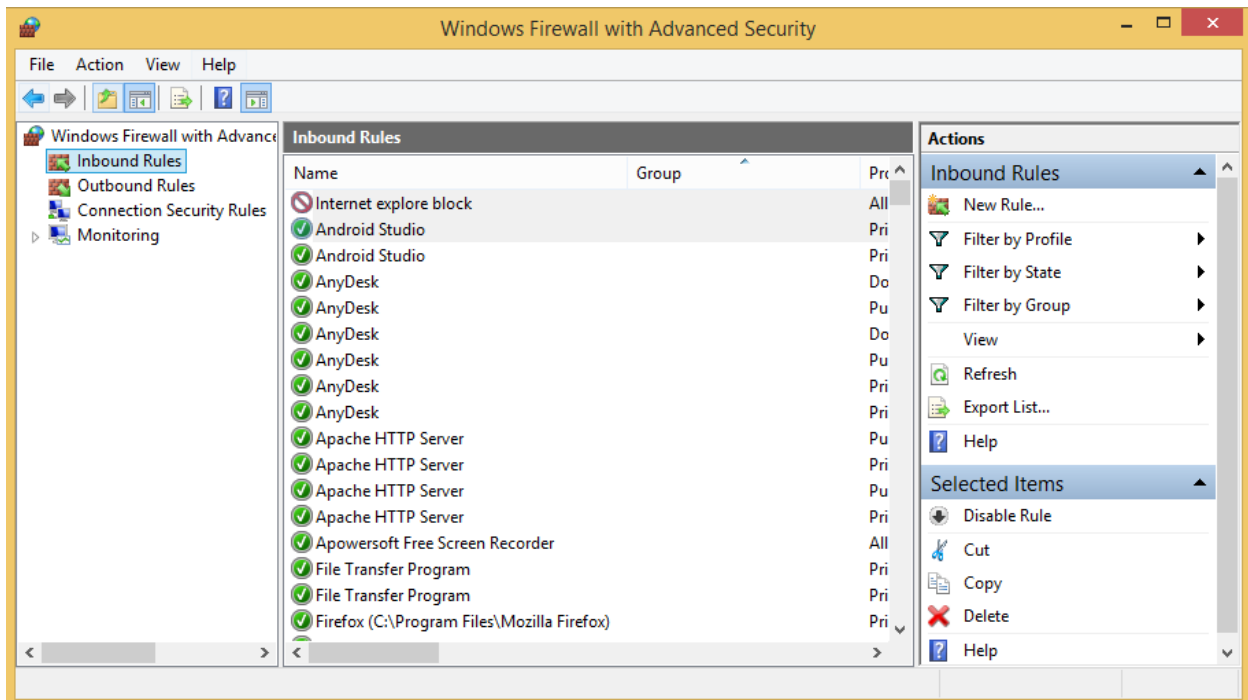
6) click on next



7) Enter the name for the rule and add description if you want to and then click on finish.

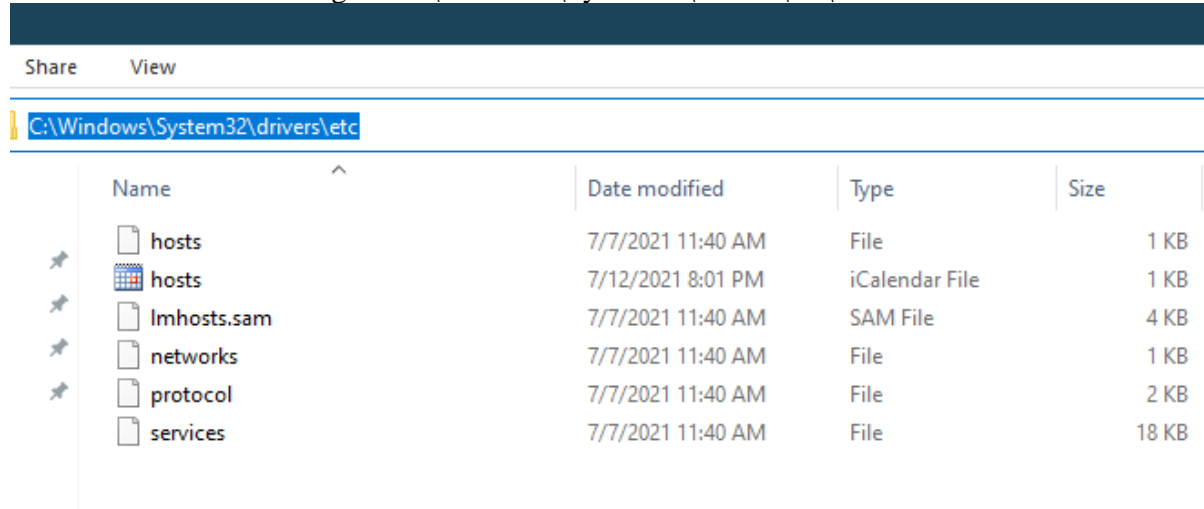


8) now Internet explore block program block now

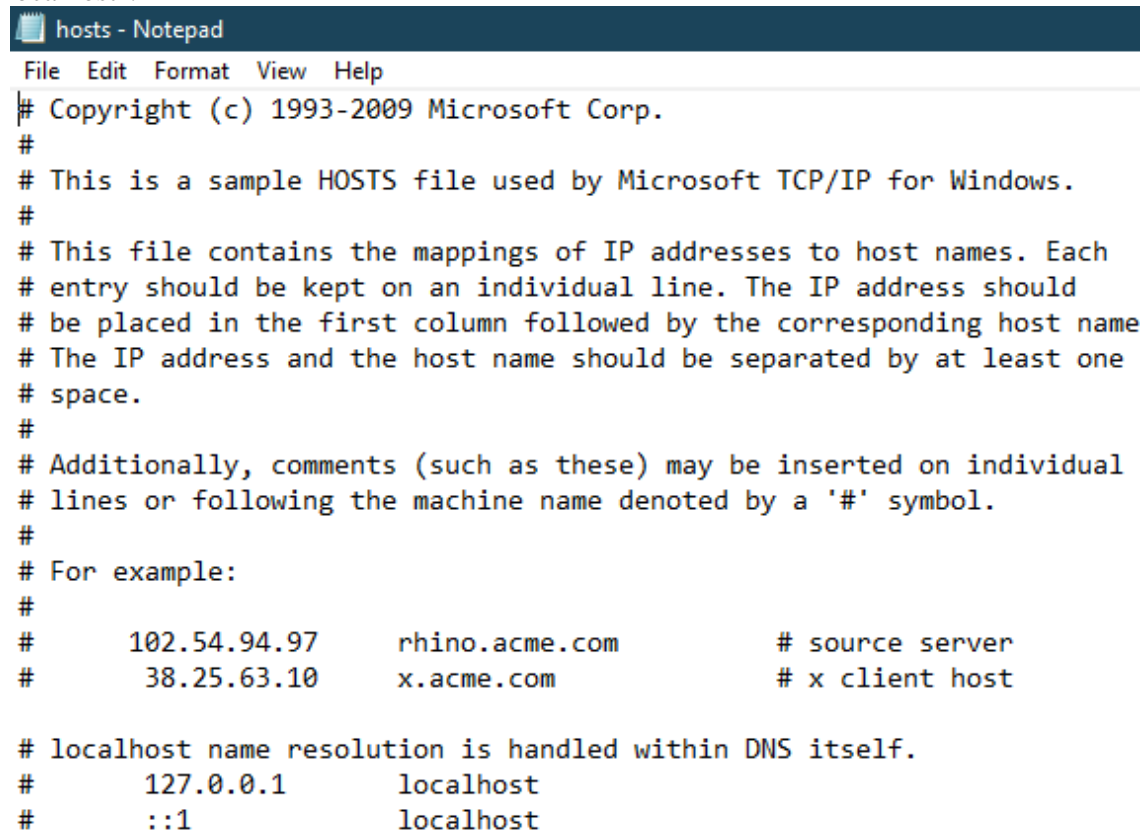


### C) -A website

1. Make sure you have administrator access on your computer. Sign in to your PC using an administrator account and go to C:\Windows\System32\drivers\etc\

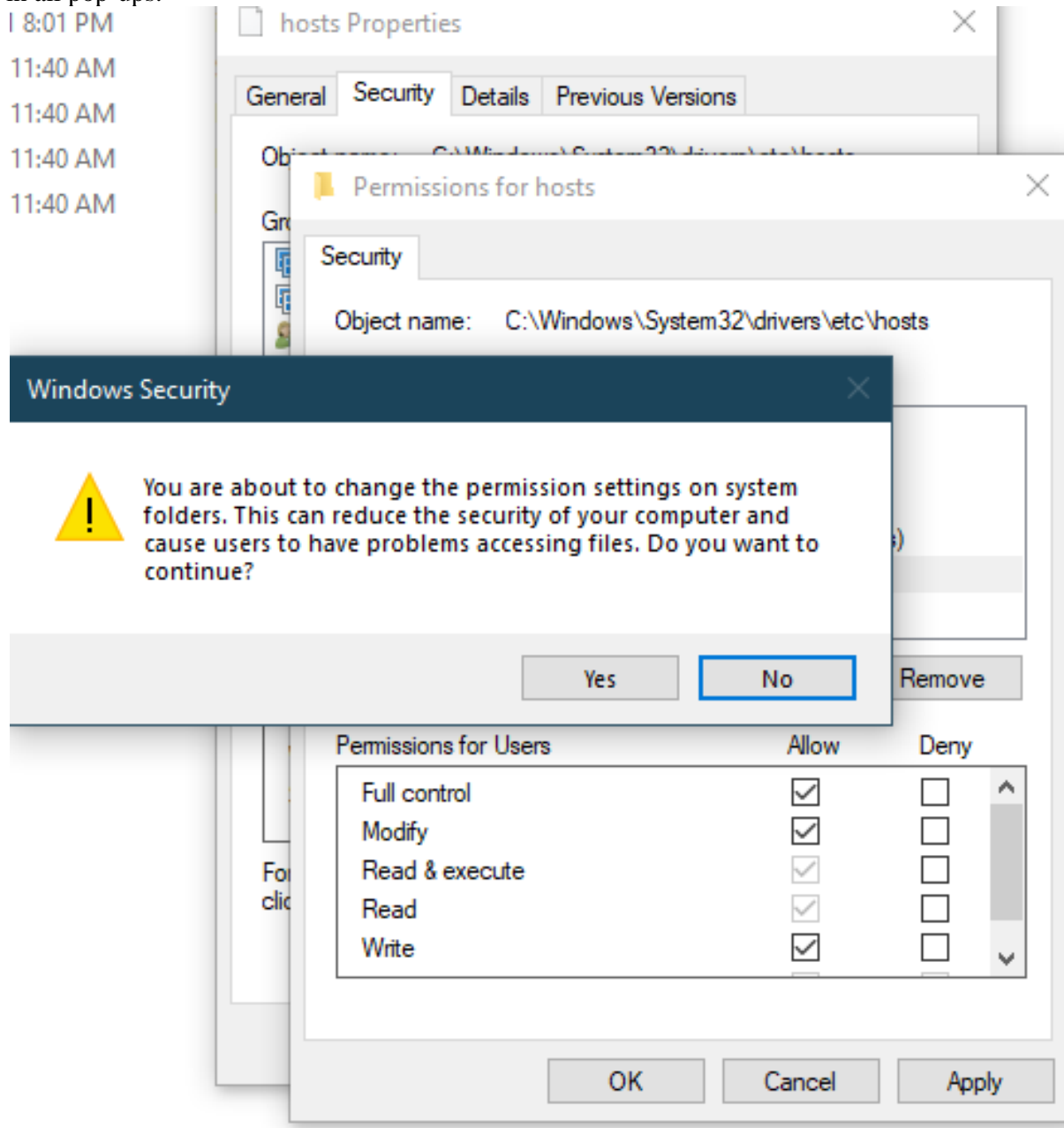


2. Double-click the file named "hosts" and select Notepad from the list of programs to open the file. Click OK. The last two lines of your hosts file should read "# 127.0.0.1 localhost" and "# ::1 localhost".

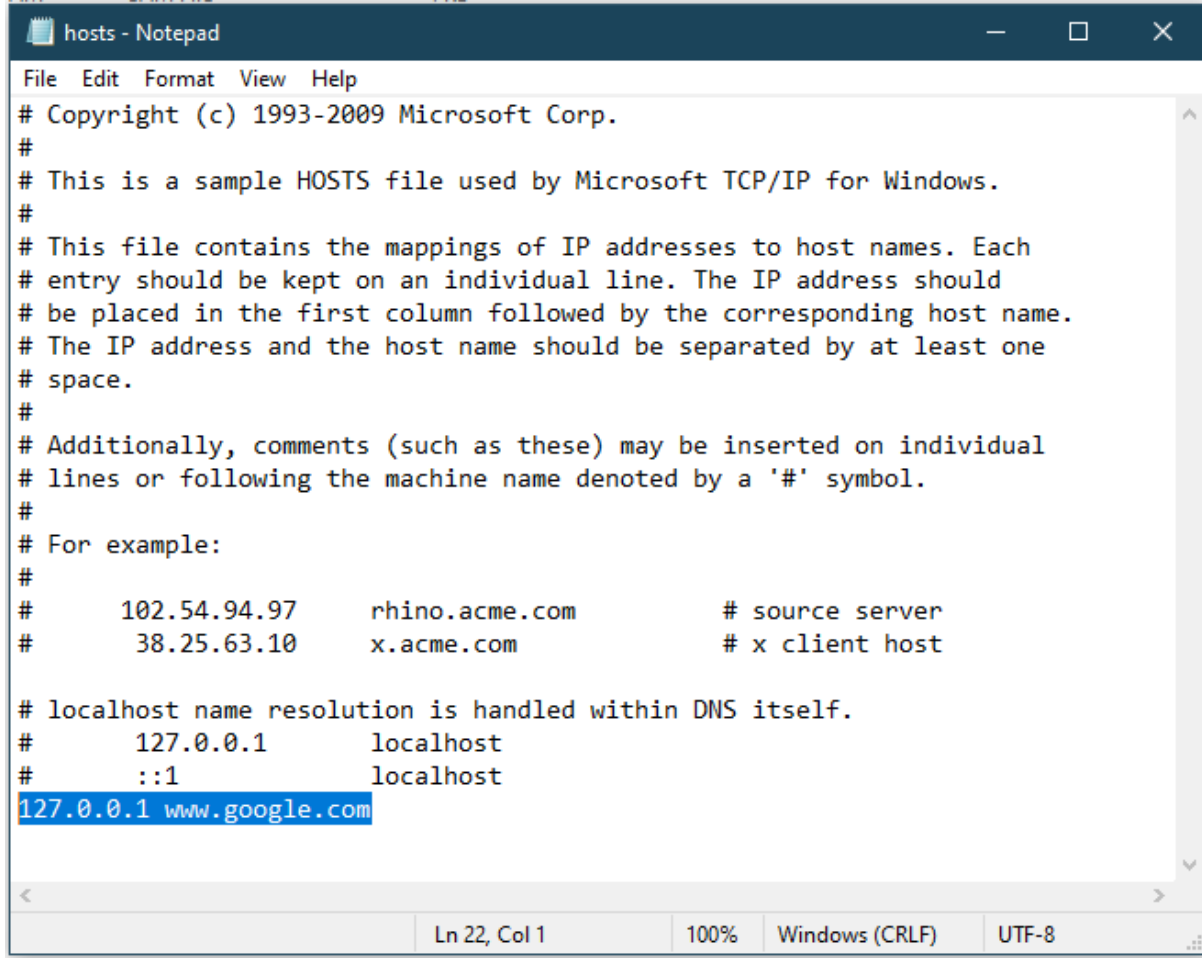


2a. In case you can't edit the file, you'll need to right-click the file labelled hosts and select Properties. Click the Security tab, select the administrator account and click Edit.

2b. In the pop-up, select the account again and check Full control. Click Apply > Yes. Now click OK in all pop-ups.



3. At the end of the file, you can add the addresses of websites to block. To do this, just add a line at the end of the file, with 127.0.0.1 and then the name of the site you want to block - this will redirect the site's name to your local computer.



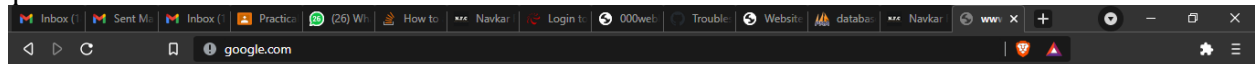
```
File Edit Format View Help
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#       102.54.94.97       rhino.acme.com          # source server
#       38.25.63.10       x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#       127.0.0.1        localhost
#       ::1              localhost
127.0.0.1 www.google.com
```

Ln 22, Col 1      100%      Windows (CRLF)      UTF-8

## INFORMATION AND NETWORK SECURITY PRACTICALS

4. To block Google, for example, add "127.0.0.1 www.google.com" to the end of the file without the quote



### This site can't be reached

www.google.com refused to connect.

Try:

- Checking the connection
- [Checking the proxy and the firewall](#)

ERR\_CONNECTION\_REFUSED

Reload

Details

