**Practical 1:**

**Aim: Write program to implement the following substitution cipher techniques:**

**1)Ceasar Cipher**

**2)Monoalphabetic Cipher**

**Caeserdemo2.java**

import java.io.\*;

class caesercipher

{

private final String Alphabet="abcdefghijklmnopqrstuvwxyz";

public String encryption(String pt,int key)

{

pt=pt.toLowerCase();

String ct="";

for(int i=0;i<pt.length();i++)

{

int charposition=Alphabet.indexOf(pt.charAt(i)); int keyval=(key+charposition)%26; char replaceval=this.Alphabet.charAt(keyval); ct=ct+replaceval;

}

return ct;

}

public String decrypt(String ct,int key)

{

ct=ct.toLowerCase();

String pt="";

for(int i=0;i<ct.length();i++)

{

int charposition=this.Alphabet.indexOf(ct.charAt(i)); int keyval=(charposition-key)%26; if(keyval<0)

{

keyval=this.Alphabet.length()+keyval;

}

char replaceval=this.Alphabet.charAt(keyval); pt=pt+replaceval;

}

return pt;

}

}

class caeserdemo2

{

public static void main(String arg[])throws IOException

{

int choice;

System.out.println("1.Encryption /n 2.Decryption");

System.out.println("Enter your Choice");

BufferedReader br1=new BufferedReader(new InputStreamReader(System.in)); choice=Integer.parseInt(br1.readLine());

System.out.println("Enter any string");

BufferedReader br=new BufferedReader(new InputStreamReader(System.in)); String pt=br.readLine(); int key=4; caesercipher cc=new caesercipher(); String ct=cc.encryption(pt,key); switch(choice)

{

case 1:

System.out.println("Plain text="+pt); System.out.println("Caeser text="+ct); break; case 2:

System.out.println("Enter any string for decrypt");

BufferedReader br2=new BufferedReader(new InputStreamReader(System.in));

String ct1=br2.readLine();

String cpt=cc.decrypt(ct,key);

System.out.println("Plain text="+cpt);

break; default:

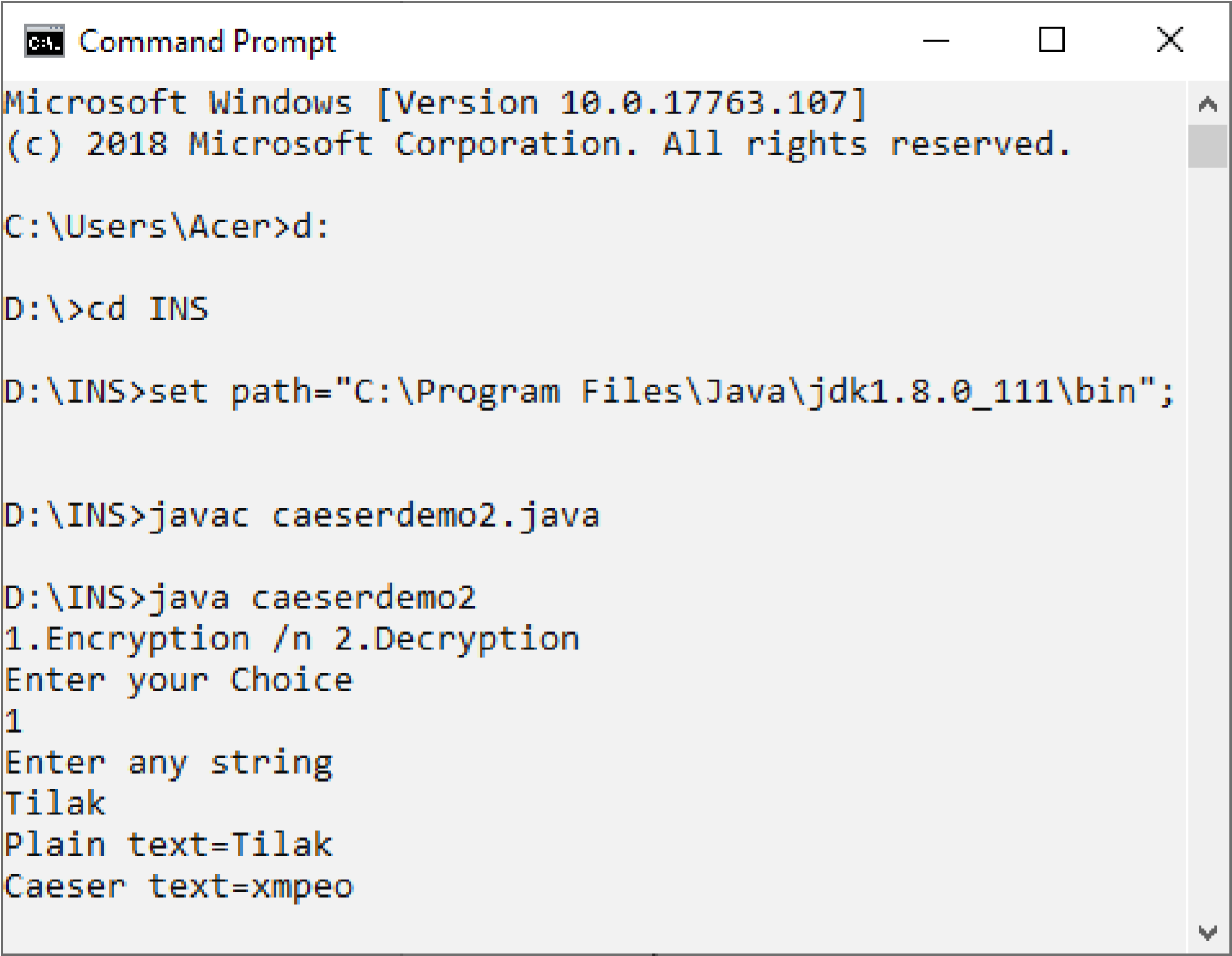
System.out.println("Wrong Choice");

}

}

}

**Output:**



**MonoAlphabeticDemo.java**

import java.io.\*;

class monoalpha

{

private final String Alphabet="abcdefghijklmnopqrstuvwxyz"; private String newkey=""; private static int isGenerated=0; private void generatedkey(String userkey)

{

userkey=userkey.toLowerCase();

for(int i=0;i<userkey.length();i++)

{

int flag=0; for(int j=0;j<this.newkey.length();j++)

{

if(userkey.charAt(i)==newkey.charAt(j))

{

flag=1; break;

}

}

if(flag==0)

this.newkey+=userkey.charAt(i);

}

if(isGenerated==0)

{

isGenerated=1; this.generatedkey(this.newkey+""+this.Alphabet);

}

}

public String encrypt(String plainText,String userkey)

{

this.generatedkey(userkey); String cipherText=""; String tmpstr=plainText; for(int i=0;i<plainText.length();i++)

{

char replaceVal=this.newkey.charAt(this.Alphabet.indexOf(plainText.charAt(i))); tmpstr=tmpstr.replace(tmpstr.charAt(i),replaceVal);

}

cipherText=tmpstr; return cipherText;

}

public String decrypt(String cipherText,String userkey)

{

this.generatedkey(userkey); String plainText=""; String tmpstr=cipherText; for(int i=0;i<cipherText.length();i++) {

char replaceVal=this.Alphabet.charAt(this.newkey.indexOf(cipherText.charAt(i))); tmpstr=tmpstr.replace(tmpstr.charAt(i),replaceVal);

}

plainText=tmpstr; return plainText;

}

}

class MonoAlphabeticDemo

{

public static void main(String args[])

{

monoalpha Ma=new monoalpha();

String en=Ma.encrypt("hihowareyou","student");

System.out.print(en);

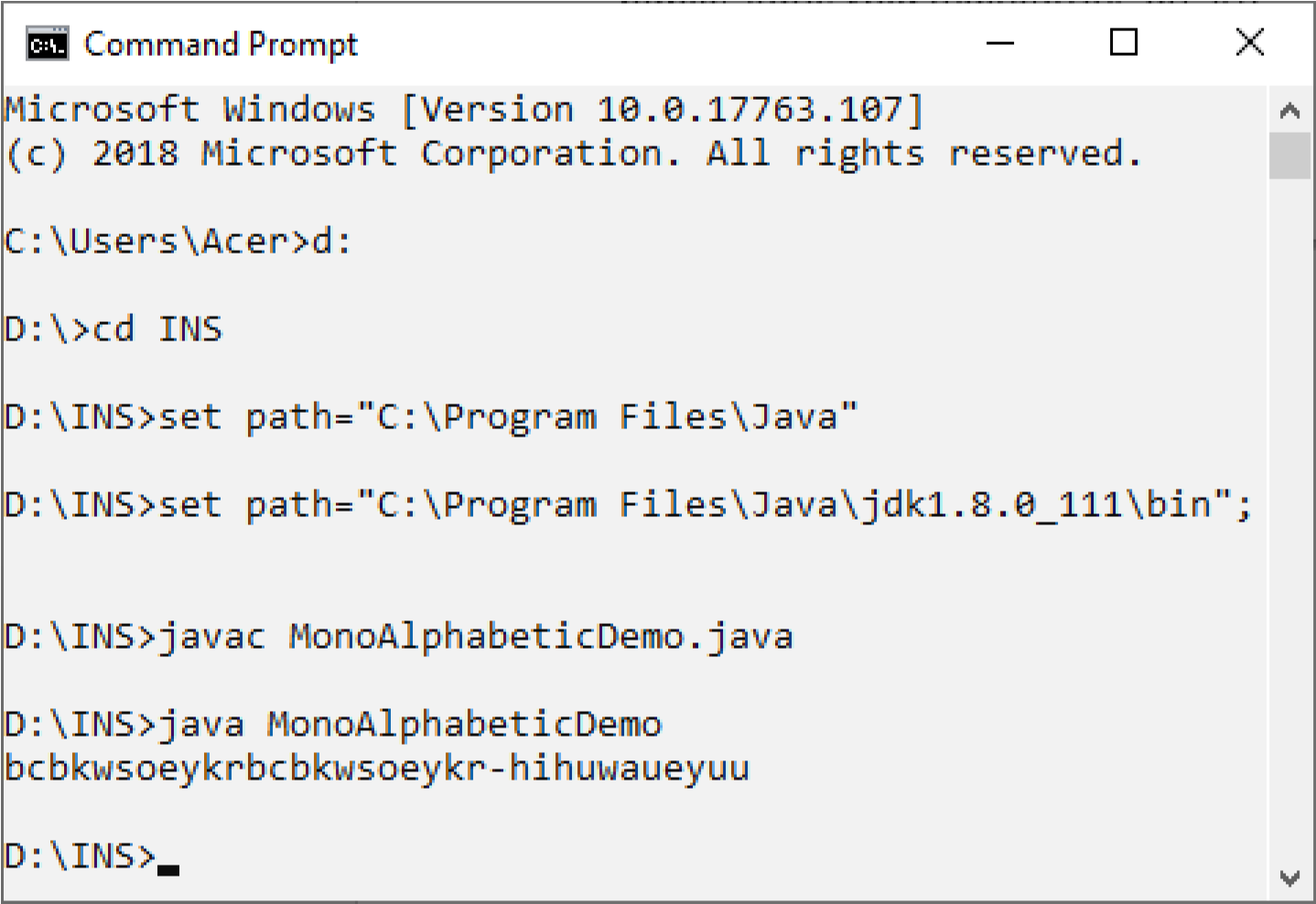
String de=Ma.decrypt(en,"student");

System.out.println(en+"-"+de);

}

}

**Output:**



**Practical 2:**

**Aim: write program to implement the following substitution cipher techniques:**

**1)Vernam Cipher**

**2)Playfair Cipher**

**Vernam.java**

import java.io.\*; class vernam

{

public static int getCharValue(char x)

{

int y=(int)'a'; return((int)x-y);

}

public static char getNumberValue(int x)

{

int z=x+(int)'a'; return ((char)z);

}

public static void main(String arg[])throws Exception

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter your plain text");

String accept=br.readLine();

System.out.println("\nEnter your one time pad text"); String pad=br.readLine(); int aval[]=new int[accept.length()]; int pval[]=new int[pad.length()]; int initval[]=new int[pad.length()]; if(pad.length()!=accept.length())

{

System.out.println("Invalid one time pad. Application terminates."); return;

}

for(int i=0;i<accept.length();i++)

{

int k=getCharValue(accept.charAt(i));

aval[i]=k;

}

for(int i=0;i<pad.length();i++)

{

int k=getCharValue(pad.charAt(i));

pval[i]=k;

}

for(int i=0;i<pad.length();i++)

{

initval[i]=aval[i]+pval[i]; if(initval[i]>25) initval[i]-=26;

}

System.out.println("\n Cipher text is:"); String cipher=""; for(int i=0;i<pad.length();i++)

{

cipher+=getNumberValue(initval[i]);

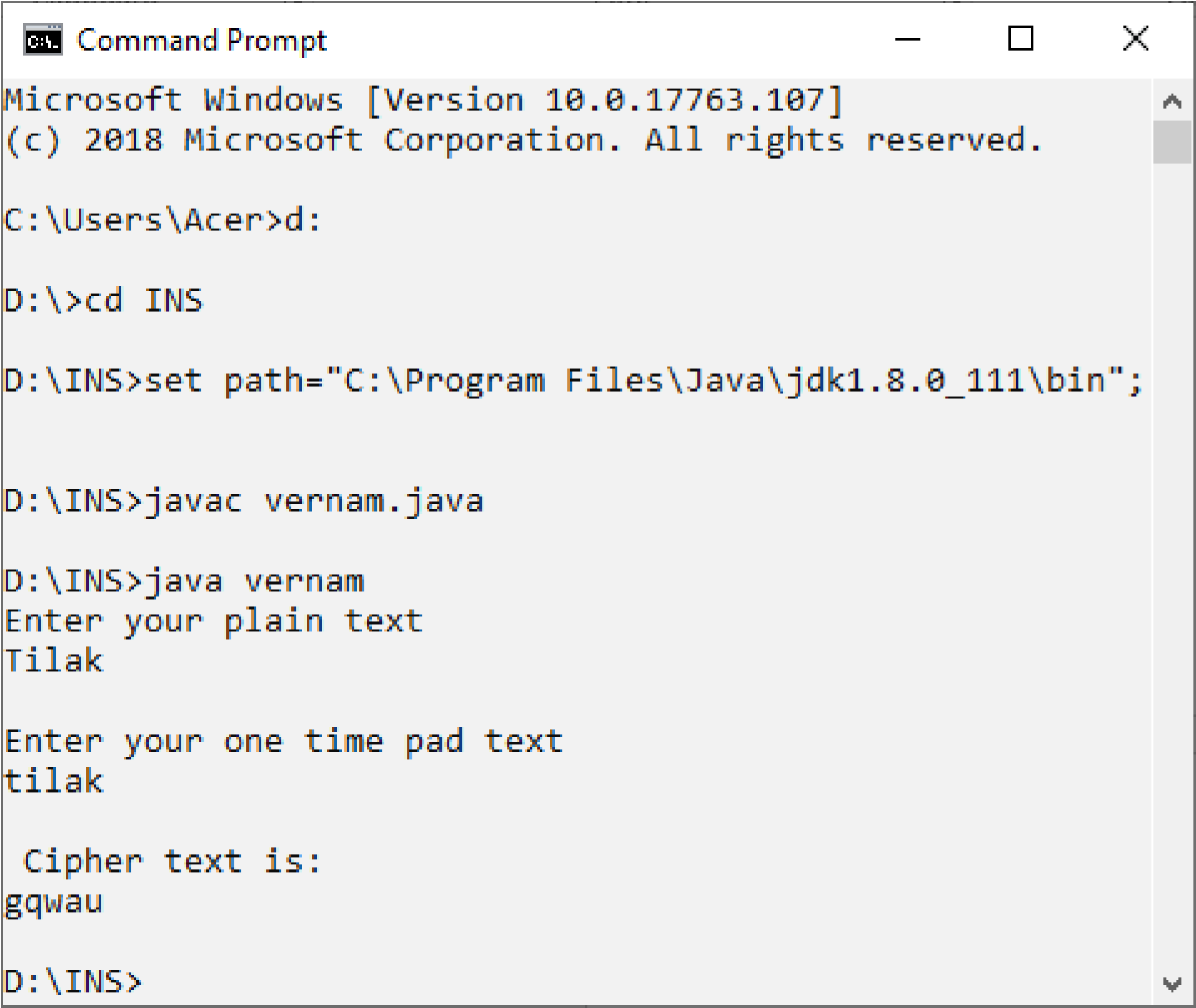
}

System.out.println(cipher);

}

}

**Output:**



**PlayFair1.java** import java.io.\*;

import java.awt.event.\*; import java.util.\*;

import java.util.Scanner;

class PlayFair1

{

public static String findIndex(String[][] arr, String test)



{

String index = "";

for(int i=0; i<arr.length; i++)

{

for(int j=0; j<arr[i].length; j++)

{

if(test.equalsIgnoreCase(arr[i][j]))

{

index = String.valueOf(i)+String.valueOf(j);

return index;

}

}

}

return null;

}

public static void main (String[] args)

{

Scanner in = new Scanner(System.in);

System.out.println("Enter the choice:");

System.out.println("e. Encrypt text."); System.out.println("d. Decrypt text."); int choice = in.nextInt();

switch(choice)

{

case 1:

findPlayfairCipher();

break;

case 2:

DecryptPlayfairCipher();

break;

default:System.out.printf("Invalid Choice"); break;

case 3:

System.out.println("Invalid choice"); break;

}

}

public static void DecryptPlayfairCipher()

{

Scanner in = new Scanner(System.in);

String plainText="",cipherText="";

String playFairMatrix[][]= {

{"H", "A", "R", "P","S"},

{"I","C", "O", "D", "B" },

{"E", "F" ,"G", "K", "L" },

{"M", "N", "Q", "T", "U"},

{"V", "W", "X", "Y","Z"},

};

System.out.println("Enter text to be decrypted:"); cipherText = in.nextLine();

for(int i=0; i<cipherText.length(); i+=2)

{

char c = cipherText.charAt(i); char d=cipherText.charAt(i); if(i+1<cipherText.length())

{

d = cipherText.charAt(i+1);

}

String val = String.valueOf(c);

String vald = String.valueOf(d);

String index1,index2;

if(val.equals(" "))

{

plainText=plainText+" ";

i--;

continue;

}

else

{

if(val.equalsIgnoreCase("J"))

{

index1 = findIndex(playFairMatrix, String.valueOf("I"));

}

else

{

index1 = findIndex(playFairMatrix, String.valueOf(cipherText.charAt(i)));

}

if(vald.equalsIgnoreCase("J"))

{

index2 = findIndex(playFairMatrix, String.valueOf("I"));

}

else

{

index2 = findIndex(playFairMatrix, String.valueOf(cipherText.charAt(i+1)));

}

if(index1.charAt(0) == index2.charAt(0))

{

int m = Integer.parseInt(String.valueOf(index1.charAt(1))); int n = Integer.parseInt(String.valueOf(index2.charAt(1))); int o = Integer.parseInt(String.valueOf(index1.charAt(0))); int p = Integer.parseInt(String.valueOf(index2.charAt(0))); if(m==0)

{

m=5;

}

if(n==0)

{

n=5;

}

plainText=plainText+playFairMatrix[o][m-1];

plainText=plainText+playFairMatrix[p][n-1];

}

else if(index1.charAt(1) == index2.charAt(1))

{

int o = Integer.parseInt(String.valueOf(index1.charAt(0))); int m = Integer.parseInt(String.valueOf(index1.charAt(1))); int p = Integer.parseInt(String.valueOf(index2.charAt(0))); int n = Integer.parseInt(String.valueOf(index2.charAt(1))); if(p==0)

{

p=5;

}

if(o==0)

{

o=5;

}

plainText=plainText+playFairMatrix[o-1][m];

plainText=plainText+playFairMatrix[p-1][n];

}

else

{

int o = Integer.parseInt(String.valueOf(index1.charAt(0))); int m = Integer.parseInt(String.valueOf(index1.charAt(1))); int p = Integer.parseInt(String.valueOf(index2.charAt(0))); int n = Integer.parseInt(String.valueOf(index2.charAt(1))); plainText=plainText+playFairMatrix[o][n]; plainText=plainText+playFairMatrix[p][m];

}

}

}

System.out.println("The decrypted text is:"); System.out.println(plainText);}

public static void findPlayfairCipher()

{

Scanner in = new Scanner(System.in);

String plainText="",plainTxt, cipherText="";

String playFairMatrix[][]=

{{"H", "A", "R", "P","S"},

{"I","C", "O", "D", "B" },

{"E", "F" ,"G", "K", "L" },

{"M", "N", "Q", "T", "U"},

{"V", "W", "X", "Y","Z"},

};

System.out.println("Enter text to be encrypted:"); plainTxt = in.nextLine(); String temp="";

String arr[]=plainTxt.split(" "); for(int j=0;j<arr.length;j++)

{

temp=arr[j]; if(temp.length()%2!=0)

{

temp=temp+"x";

}

plainText=plainText+ temp+" ";

}

for(int i=0; i<plainText.length(); i+=2)

{

char c = plainText.charAt(i); char d=plainText.charAt(i); if(i+1<plainText.length())

{

d = plainText.charAt(i+1);

}

String val = String.valueOf(c);

String vald = String.valueOf(d); String index1,index2;

if(val.equals(" "))

{

cipherText=cipherText+" ";

i--;

continue;

}

else

{

if(val.equalsIgnoreCase("J"))

{

index1 = findIndex(playFairMatrix, String.valueOf("I"));

}

else

{

index1 = findIndex(playFairMatrix, String.valueOf(plainText.charAt(i)));

}

if(vald.equalsIgnoreCase("J"))

{

index2 = findIndex(playFairMatrix, String.valueOf("I"));

}

else{

index2 = findIndex(playFairMatrix, String.valueOf(plainText.charAt(i+1)));

}

if(index1.charAt(0) == index2.charAt(0))

{

int m = Integer.parseInt(String.valueOf(index1.charAt(1))); int n = Integer.parseInt(String.valueOf(index2.charAt(1))); int o = Integer.parseInt(String.valueOf(index1.charAt(0))); int p = Integer.parseInt(String.valueOf(index2.charAt(0))); if(m==4)

{

m=-1;

} if(n==4)

{

n=-1;

}

cipherText=cipherText+playFairMatrix[o][m+1]; cipherText=cipherText+playFairMatrix[p][n+1];

}

else if(index1.charAt(1) == index2.charAt(1))

{

int o = Integer.parseInt(String.valueOf(index1.charAt(0))); int m = Integer.parseInt(String.valueOf(index1.charAt(1))); int p = Integer.parseInt(String.valueOf(index2.charAt(0))); int n = Integer.parseInt(String.valueOf(index2.charAt(1))); if(p>3)

{

p=-1;

}

if(o>3)

{

o=-1;

}

cipherText=cipherText+playFairMatrix[o+1][m];

cipherText=cipherText+playFairMatrix[p+1][n];

}

Else

{

int o = Integer.parseInt(String.valueOf(index1.charAt(0))); int m = Integer.parseInt(String.valueOf(index1.charAt(1))); int p = Integer.parseInt(String.valueOf(index2.charAt(0))); int n = Integer.parseInt(String.valueOf(index2.charAt(1))); cipherText=cipherText+playFairMatrix[o][n]; cipherText=cipherText+playFairMatrix[p][m];

}

}

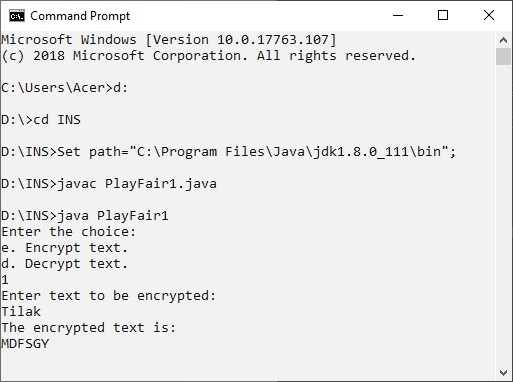
}

System.out.println("The encrypted text is:");

System.out.println(cipherText);

} }

**Output:**



**Practical 3:**

**Aim: Write program to implement the following transposition cipher techniques:**

**1)Railfence Cipher**

**2)Simple Columner Technique**

**Railfence.java** import java.io.\*; import java.awt.event.\*; import java.util.\*; public class railfence

{

public static void main(String args[])

{

String input="hellotycs"; String output=""; int len=input.length(),flag=0;

System.out.println("input string="+input); for(int i=0;i<len;i+=2)

{

output+=input.charAt(i);

}

for(int i=1;i<len;i+=2)

{

output+=input.charAt(i);

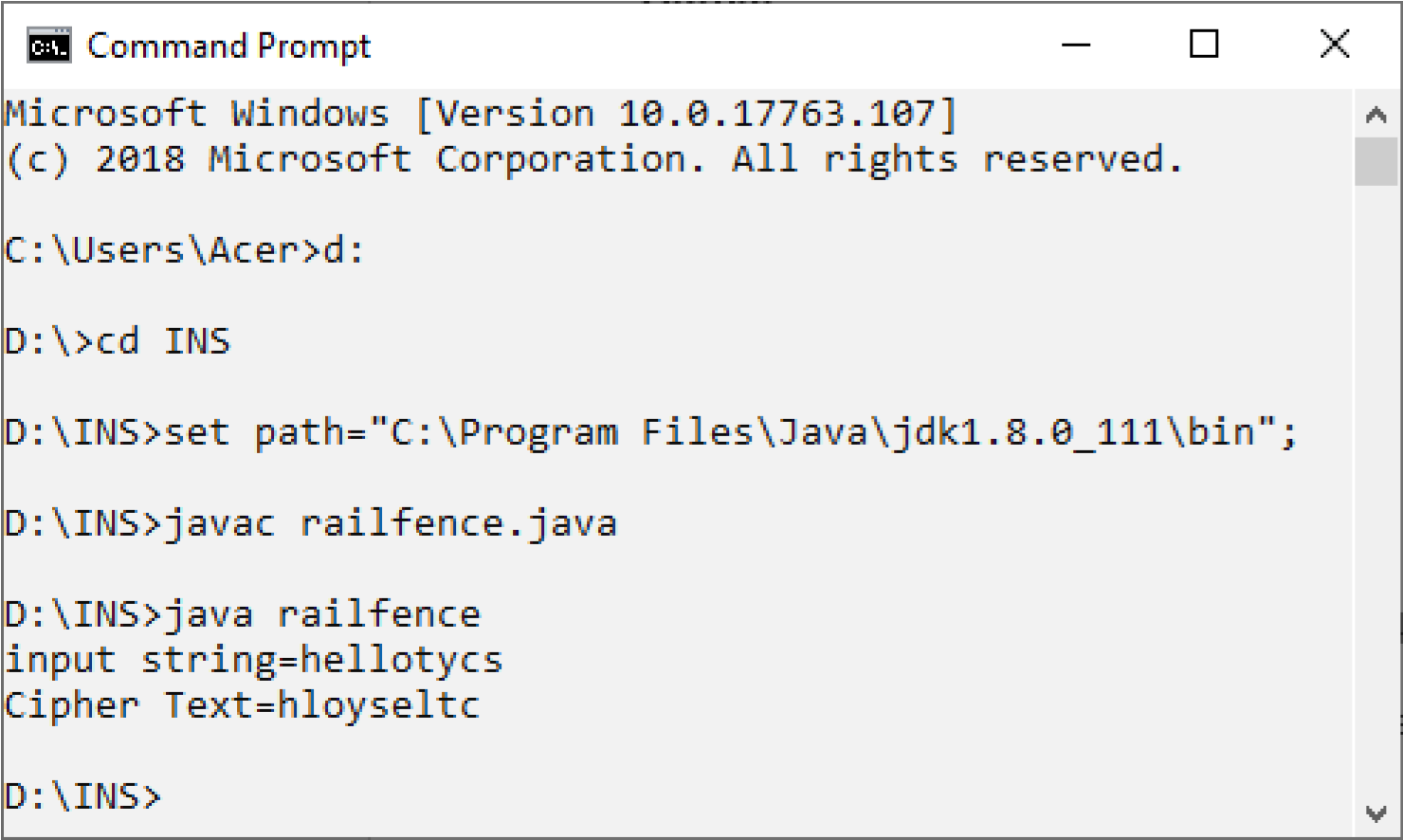
}

System.out.println("Cipher Text="+output);

}

}

**Output:**



**Columnar.java**

import java.io.\*; public class columnar

{

char arr[][],encrypt[][],decrypt[][],keya[],keytemp[]; public void createMatrix(String s,String key,int row,int column) {

arr=new char[row][column];

int k=0;

keya=key.toCharArray();

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

{

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

{

if(k<s.length())

{

arr[i][j]=s.charAt(k); k++;

}

else

{ arr[i][j]=' ';

} } } }

public void createkey(String key,int column)

{

keytemp=key.toCharArray();

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

{

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

{

if(keytemp[i]>keytemp[j])

{

char temp=keytemp[i]; keytemp[i]=keytemp[j]; keytemp[j]=temp;

} } } }

public void createMatrixD(String s,String key,int row,int column)

{

arr=new char[row][column];

int k=0;

keya=key.toCharArray(); for(int i=0;i<column;i++)

{

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

{

if(k<s.length())

{

arr[j][i]=s.charAt(k); k++;

}

else

{

arr[j][i]=' ';

} } } }

public void encrypt(int row,int column)

{

encrypt=new char[row][column];

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

{

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

{

if(keya[i]==keytemp[j])

{

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

{

encrypt[k][j]=arr[k][i];

}

keytemp[j]='?'; break;

} } } }

public void decrypt(int row,int column)

{

decrypt=new char[row][column];

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

{

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

{

if(keya[j]==keytemp[i])

{

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

{

decrypt[k][j]=arr[k][i];

}

keya[j]='?'; break;

}

}

}

}

public void resultE(int row,int column,char arr[][])

{

System.out.println("Result = "); for(int i=0;i<column;i++)

{

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

{

System.out.println(arr[j][i]);

}

}

}

public void resultD(int row,int column,char arr[][])

{

System.out.println("Result = "); for(int i=0;i<row;i++)

{

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

{

System.out.println(arr[j][i]);

}

}

}

public static void main(String args[])throws IOException

{

int row,column,choice; columnar obj=new columnar();

BufferedReader in =new BufferedReader(new InputStreamReader(System.in)); System.out.println("Menu\n 1 Encryption\n 2 Decryption"); choice=Integer.parseInt(in.readLine());

System.out.println("Enter the string ");

String s=in.readLine();

System.out.println("Enter the key ");

String key=in.readLine(); row=s.length()/key.length(); if(s.length()%key.length()!=0) row++; column=key.length(); switch(choice)

{

case 1:

obj.createMatrix(s,key,row,column); obj.createkey(key,column); obj.encrypt(row,column); obj.resultE(row,column,obj.encrypt); break; case 2:

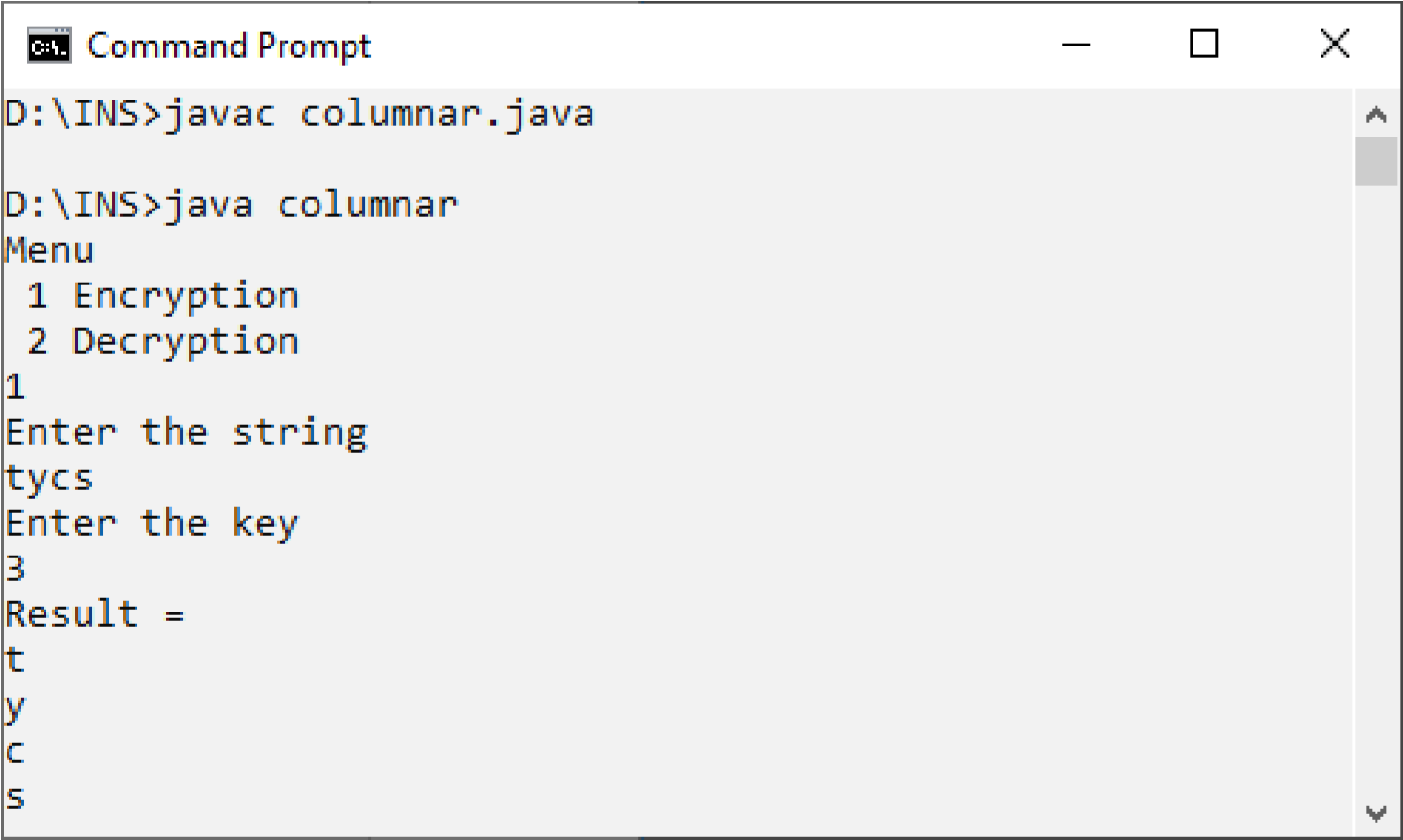
obj.createMatrixD(s,key,row,column); obj.createkey(key,column); obj.decrypt(row,column); obj.resultD(row,column,obj.decrypt); break;

}

}

}

**Output:**



**Practical 4:**

**Aim: write program to encrypt and decrypt string using**

**1)DES**

**2)AES**

**DES.java**

i import java.io.\*; import javax.crypto.Cipher; import javax.crypto.KeyGenerator; import javax.crypto.SecretKey; import sun.misc.BASE64Encoder; import sun.misc.BASE64Decoder;

public class DES

{

public static void main(String args[])throws Exception

{

String pt,ct; SecretKey Key; pt="Hello";

Key=KeyGenerator.getInstance("DES").generateKey(); ct=doEncrypt(pt,Key); System.out.print(ct); pt=doDecrypt(ct,Key);

System.out.println(pt);

}

static String doEncrypt(String pt,SecretKey Key)throws Exception

{

Cipher C=Cipher.getInstance("DES"); C.init(Cipher.ENCRYPT\_MODE,Key); byte[]ptBytes=pt.getBytes("UTF8"); byte[]enc=C.doFinal(ptBytes);

String str=new BASE64Encoder().encode(enc);

return str;

}

static String doDecrypt(String ct,SecretKey Key)throws Exception

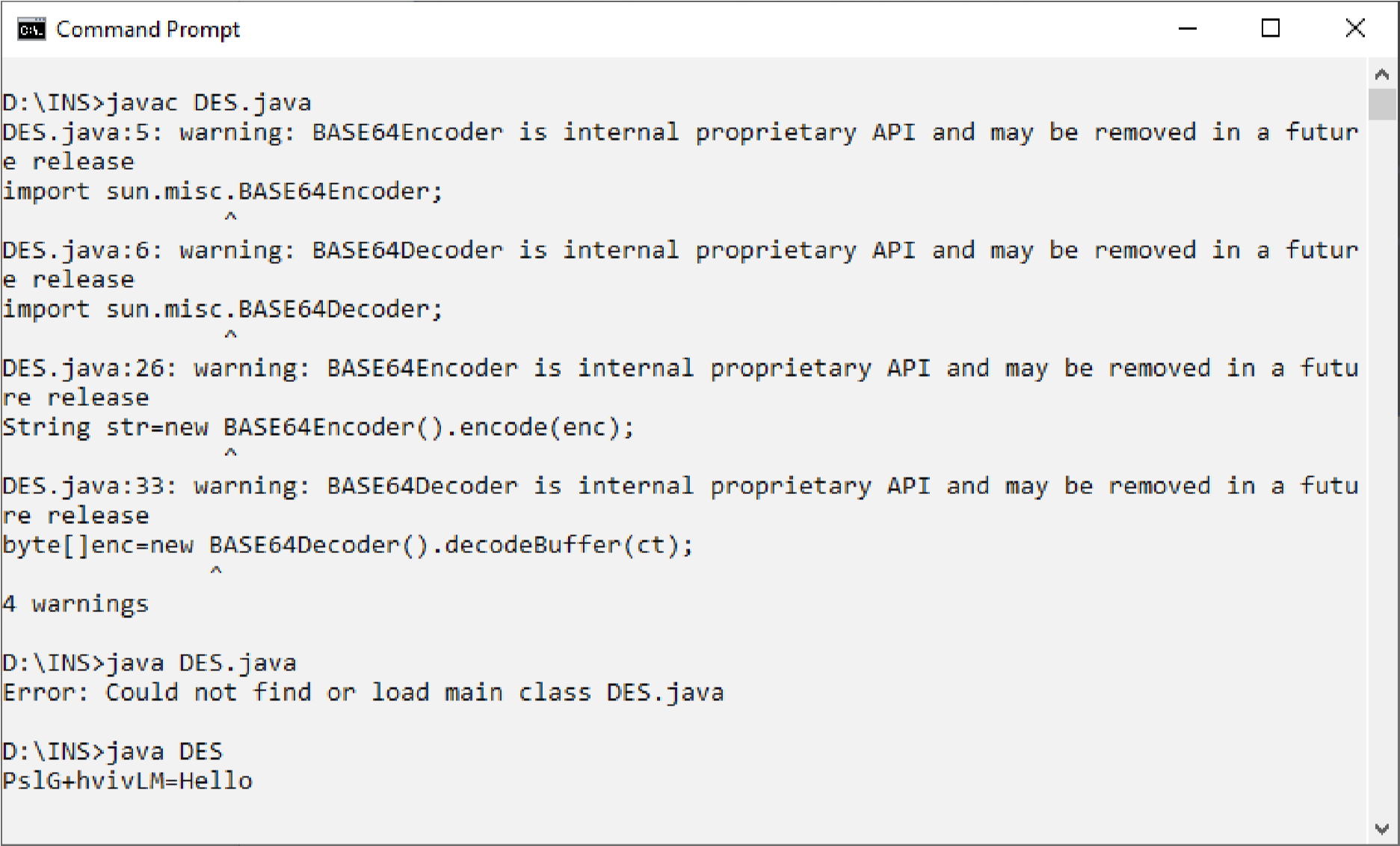
{

Cipher C=Cipher.getInstance("DES"); C.init(Cipher.DECRYPT\_MODE,Key); byte[]enc=new BASE64Decoder().decodeBuffer(ct); byte[]ptBytes=C.doFinal(enc); String str=new String(ptBytes,"UTF8"); return str;

}

}

**Output:**



**AES.java** import java.io.\*; import javax.crypto.Cipher; import javax.crypto.KeyGenerator; import javax.crypto.SecretKey; import sun.misc.BASE64Encoder; import sun.misc.BASE64Decoder;

public class AES { public static void main(String args[])throws Exception { String pt,ct; SecretKey Key; pt="Hello";

Key=KeyGenerator.getInstance("AES").generateKey(); ct=doEncrypt(pt,Key); System.out.println(ct); pt=doDecrypt(ct,Key);

System.out.println(pt);

}

static String doEncrypt(String pt,SecretKey Key)throws Exception {

Cipher C=Cipher.getInstance("AES"); C.init(Cipher.ENCRYPT\_MODE,Key); byte[]ptBytes=pt.getBytes("UTF8"); byte[]enc=C.doFinal(ptBytes);

String str=new BASE64Encoder().encode(enc);

return str;

}

static String doDecrypt(String ct,SecretKey Key)throws Exception {

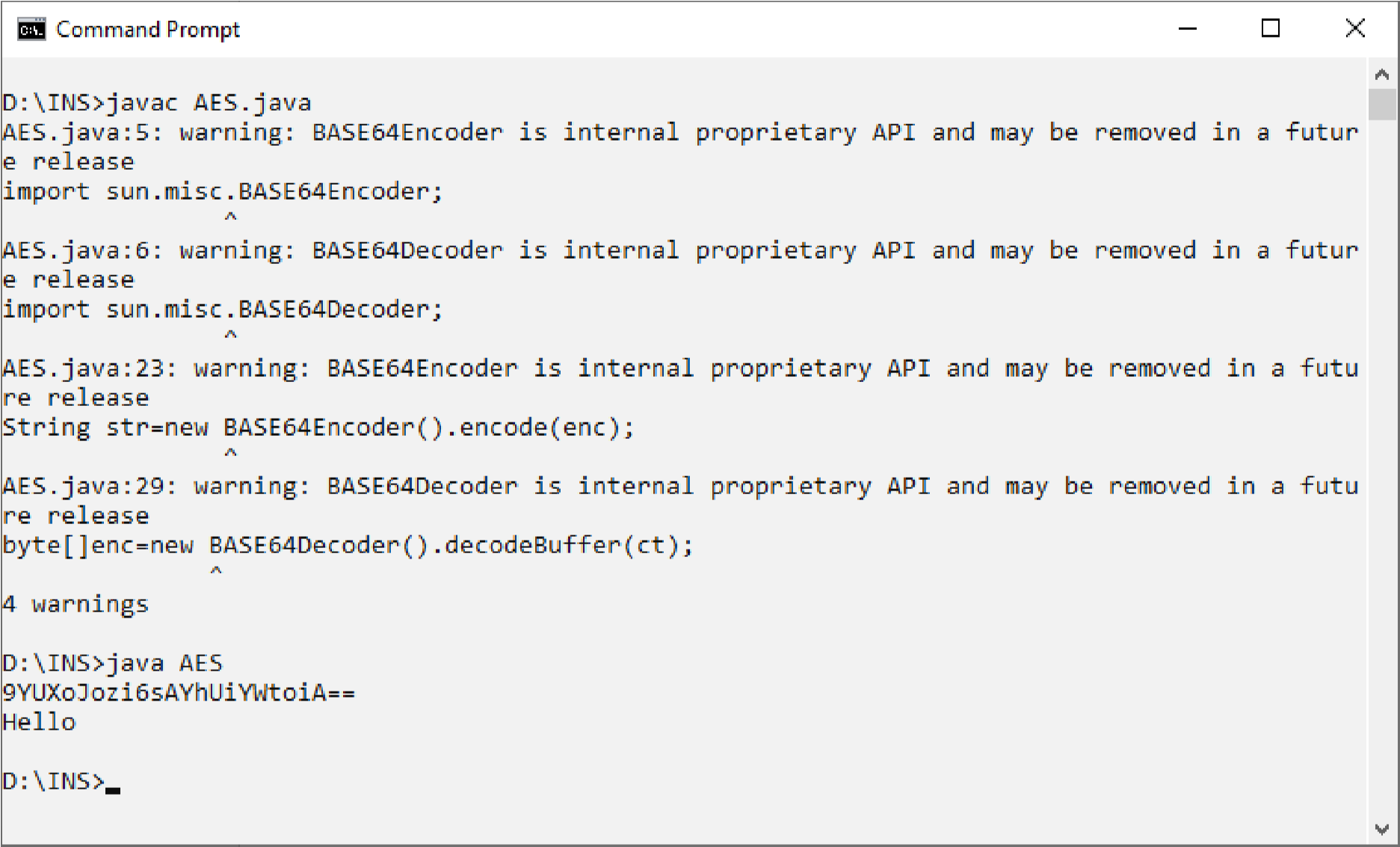
Cipher C=Cipher.getInstance("AES"); C.init(Cipher.DECRYPT\_MODE,Key); byte[]enc=new BASE64Decoder().decodeBuffer(ct); byte[]ptBytes=C.doFinal(enc);

String str=new String(ptBytes,"UTF8");

return str;

} }

**Output:**



**Practical 5:**

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

**RSA.java** import java.io.DataInputStream; import java.io.IOException; import java.math.BigInteger; import java.util.Random; public class RSA

{

private BigInteger pa; private BigInteger qb; private BigInteger nc; private BigInteger phir; private BigInteger eh; private BigInteger dc; private int b1=1024; private Random r1; public RSA()

{

r1=new Random(); pa=BigInteger.probablePrime(b1,r1); qb=BigInteger.probablePrime(b1,r1); nc=pa.multiply(qb); phir=pa.subtract(BigInteger.ONE).multiply(qb.subtract(BigInteger.ONE)); eh=BigInteger.probablePrime(b1/2,r1); while(phir.gcd(eh).compareTo(BigInteger.ONE)>0&&eh.compareTo(phir)<0)

{

eh.add(BigInteger.ONE);

}

dc=eh.modInverse(phir);

}

public RSA(BigInteger eh,BigInteger dc,BigInteger nc)

{

this.eh=eh; this.dc=dc; this.nc=nc;

}

public static void main(String args[])throws IOException

{

RSA rsa=new RSA();

DataInputStream in=new DataInputStream(System.in);

String ts;

System.out.println("Enter the plain text: "); ts=in.readLine();

System.out.println("Encrypted string :"+ts);

System.out.println("String in bytes:"+bytesToString(ts.getBytes())); byte[] encrypt=rsa.encrypt(ts.getBytes()); byte[] decrypt=rsa.decrypt(encrypt);

System.out.println("String in bytes:"+bytesToString(decrypt));

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

}

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(eh,nc).toByteArray();

}

public byte[] decrypt(byte[] message)

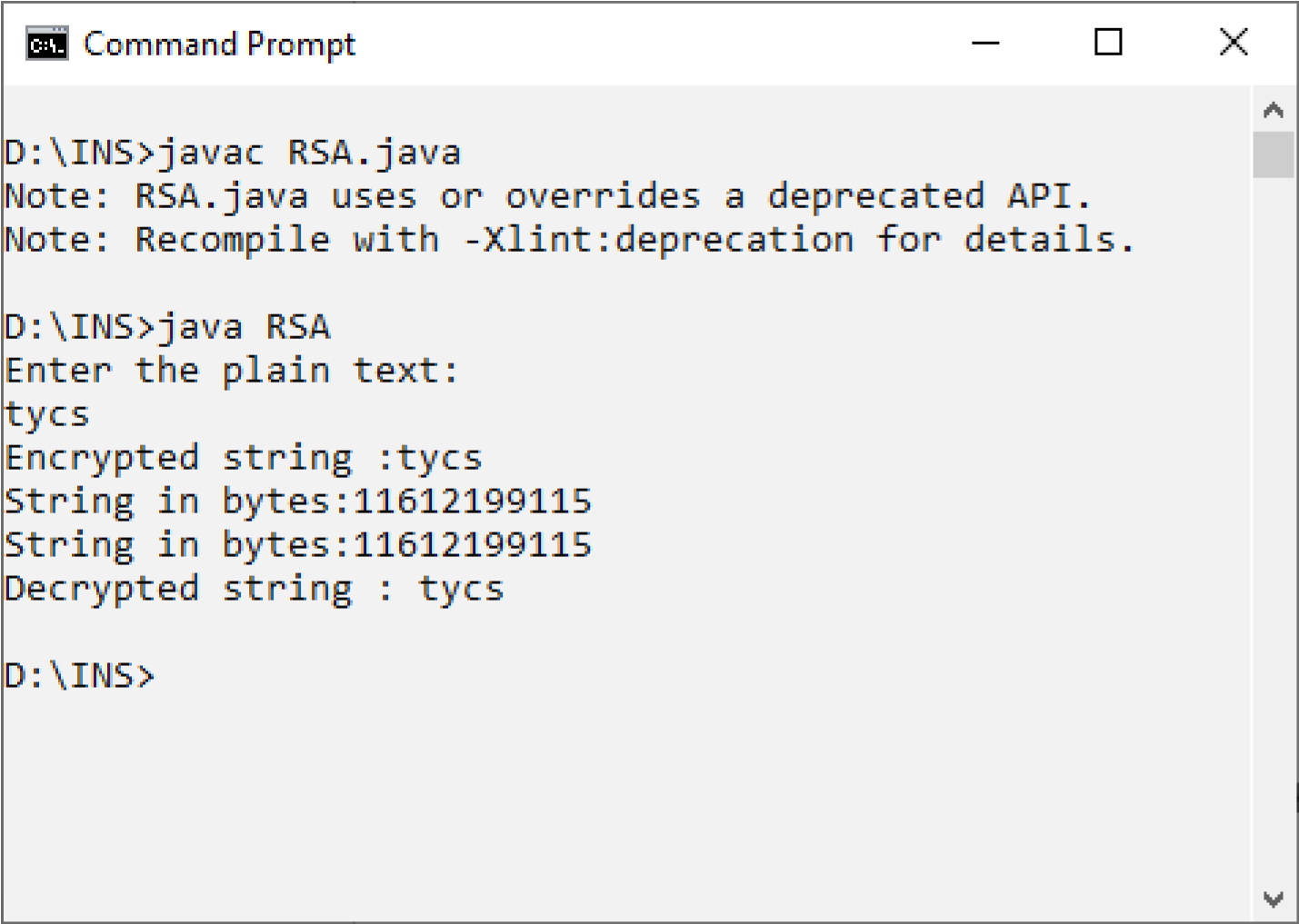
{

return(new BigInteger(message)).modPow(dc,nc).toByteArray();

}

}

**Output:**



**Practical 6:**

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

**Dh.java** import java.util.Scanner; import java.math.BigInteger; public class Dh

{

public static void main(String[] args)

{

Scanner stdin=new Scanner(System.in);

BigInteger n,g,x,y,k1,k2,A,B;

System.out.println("Enter two prime numbers"); n=new BigInteger(stdin.next()); g=new BigInteger(stdin.next());

System.out.println("Person A:Enter your secret number"); x=new BigInteger(stdin.next());

A=g.modPow(x,n);

System.out.println("Person B:Enter your secret number"); y=new BigInteger(stdin.next()); B=g.modPow(y,n); k1=B.modPow(x,n); k2=A.modPow(y,n);

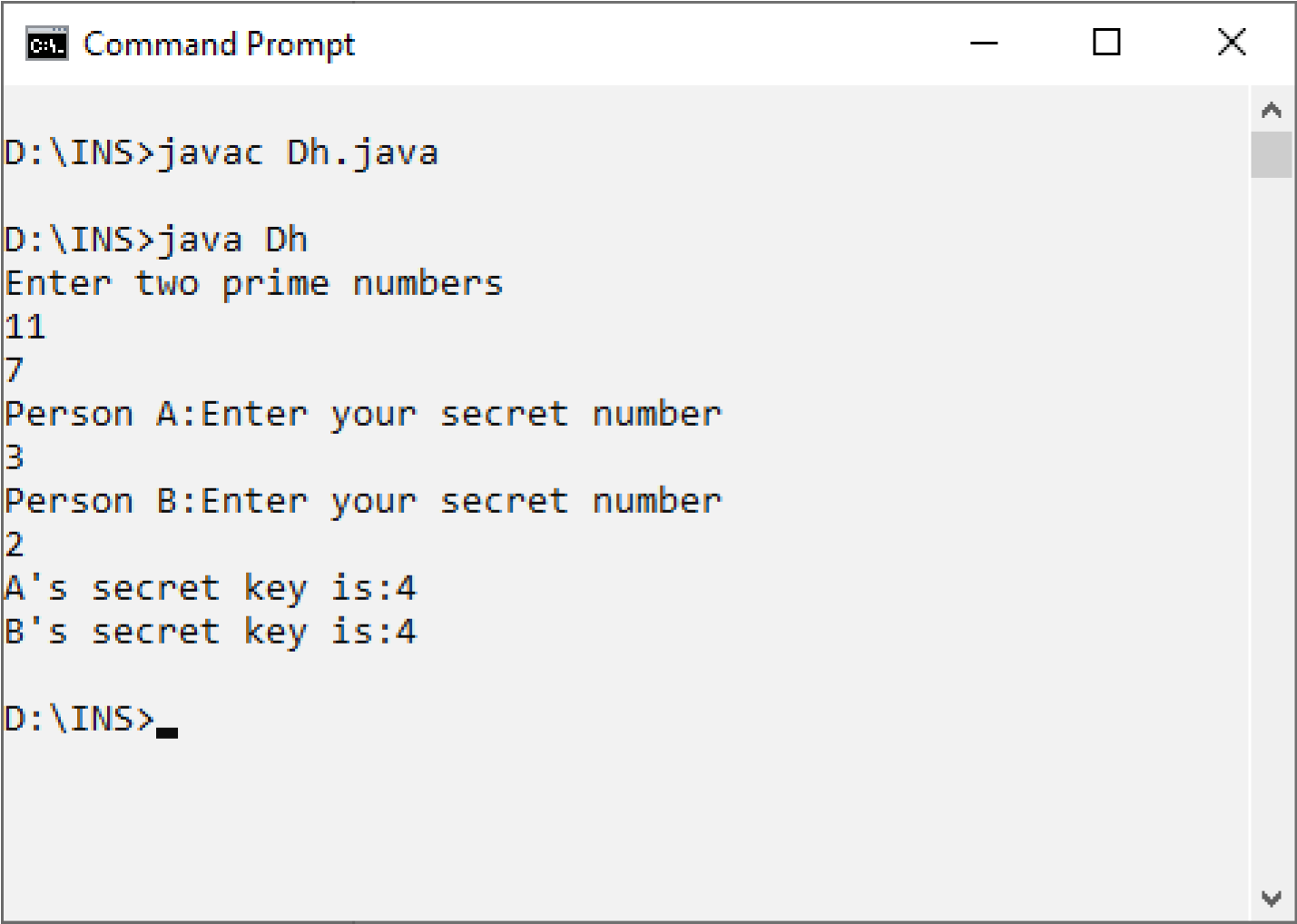
System.out.println("A's secret key is:"+k1);

System.out.println("B's secret key is:"+k2);

}

}

**Output:**



**Practical 7:**

**Aim: Write a program to implements the MD5 algorithm compute the message digest.JavaMD5Hash.java** import java.math.BigInteger; import java.security.MessageDigest; import java.security.NoSuchAlgorithmException; public class JavaMD5Hash

{

public static void main(String[] args) {

System.out.println("For null " + md5(""));

System.out.println("For simple text "+ md5("This is my text"));

System.out.println("For simple numbers " + md5("12345"));

}

public static String md5(String input) { String md5 = null; if(null == input) return null;

try {

MessageDigest digest = MessageDigest.getInstance("MD5"); digest.update(input.getBytes(), 0, input.length()); md5 = new BigInteger(1, digest.digest()).toString(16);

}

catch (NoSuchAlgorithmException e) {

e.printStackTrace();

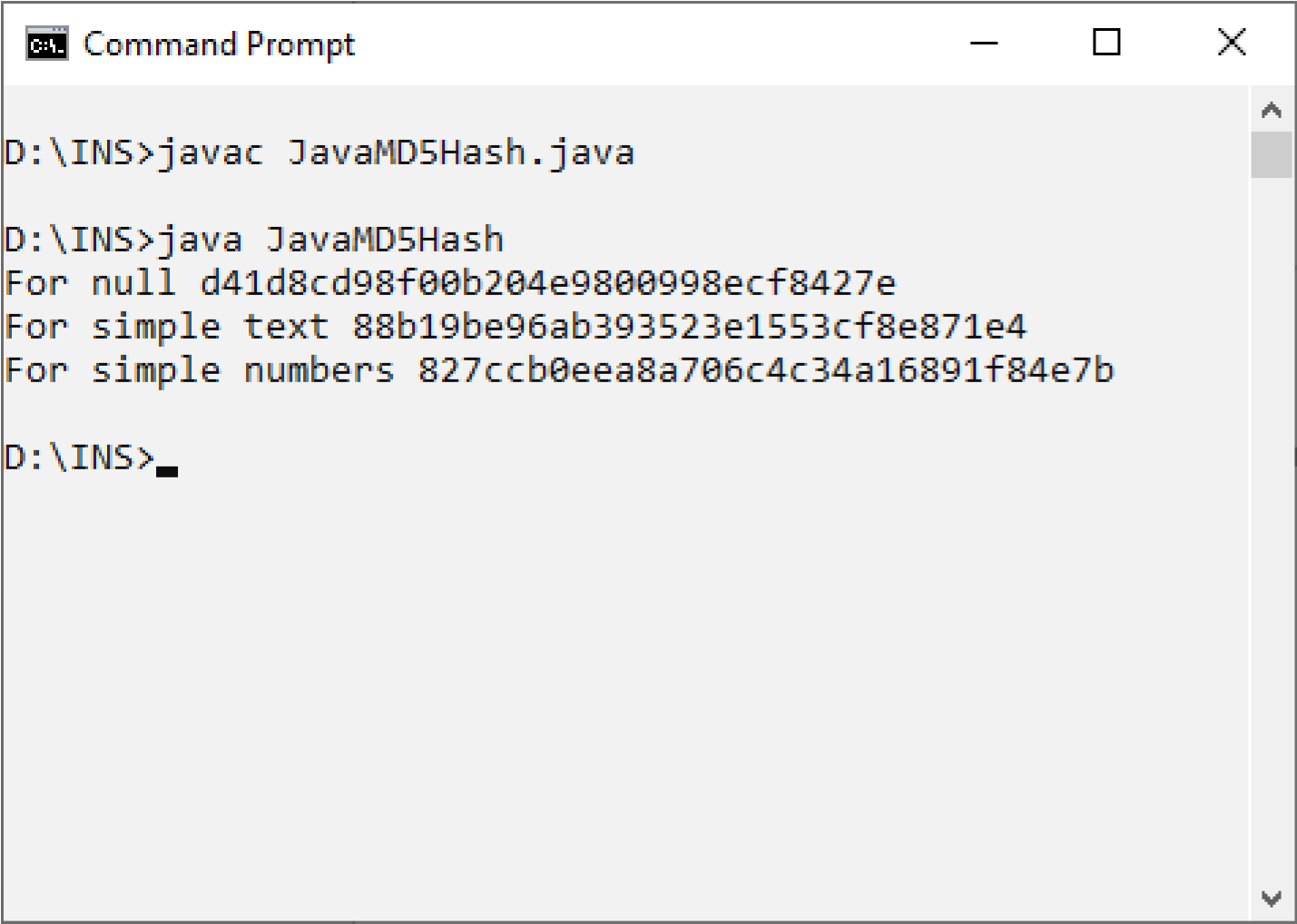
}

return md5;

}

}

**Output:**



**Practical 8:**

**Aim: Write a program to calculate the HMAC-SHA1 signature.**

**HmacSha1Signature.java**

import java.security.InvalidKeyException; import java.security.NoSuchAlgorithmException; import java.security.SignatureException; import java.util.Formatter; import javax.crypto.Mac; import javax.crypto.spec.SecretKeySpec; public class HmacSha1Signature { private static final String HMAC\_SHA1\_ALGORITHM = "HmacSHA1";

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

for (byte b : bytes) {

formatter.format("%02x", b);

}

return formatter.toString();

}

public static String calculateRFC2104HMAC(String data, String key) throws SignatureException, NoSuchAlgorithmException, InvalidKeyException

{

SecretKeySpec signingKey = new SecretKeySpec(key.getBytes(), HMAC\_SHA1\_ALGORITHM); Mac mac = Mac.getInstance(HMAC\_SHA1\_ALGORITHM);

mac.init(signingKey); return toHexString(mac.doFinal(data.getBytes()));

}

public static void main(String[] args) throws Exception {

String hmac = calculateRFC2104HMAC("data", "key"); System.out.println(hmac); assert hmac.equals("104152c5bfdca07bc633eebd46199f0255c9f49d");

}

}

**Output:**



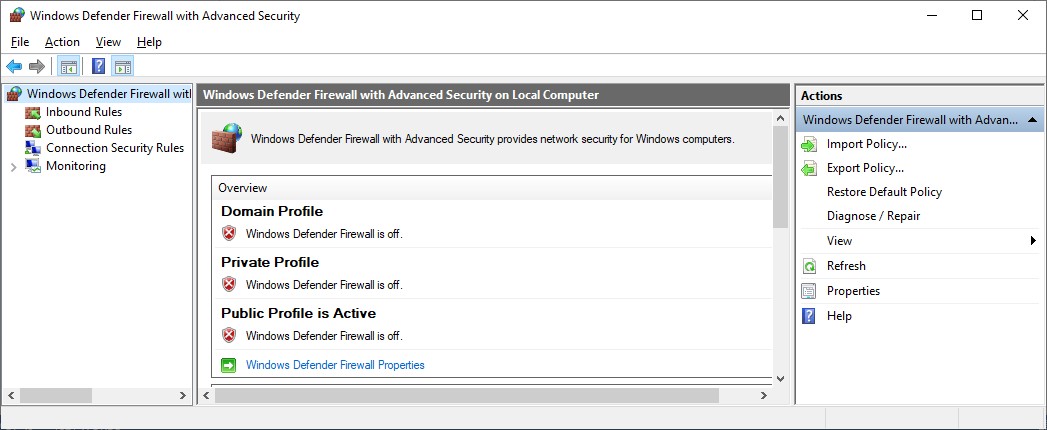
**Practical 9:**

**Aim:Configure Windows Firewall to Block:**

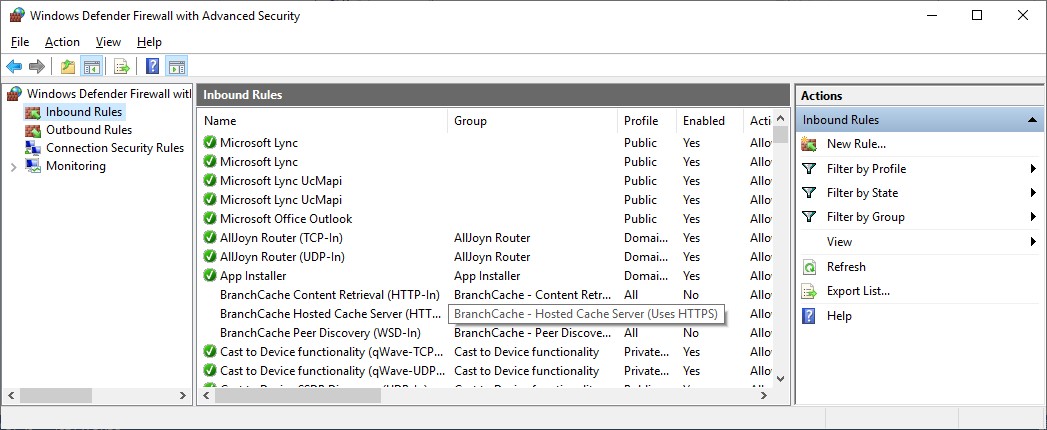
* **A Port**
* **A Program**
* **A Website**

**For a Port**

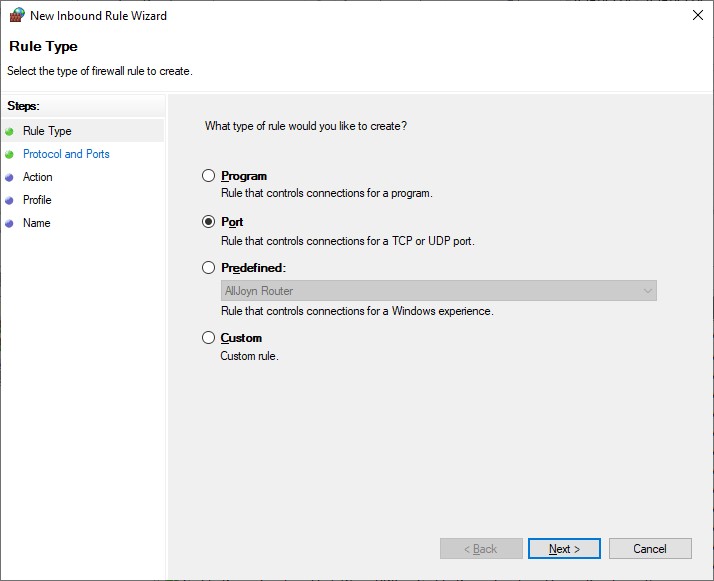
**1.Search Windows Firewall with advanced setting and Open it.**



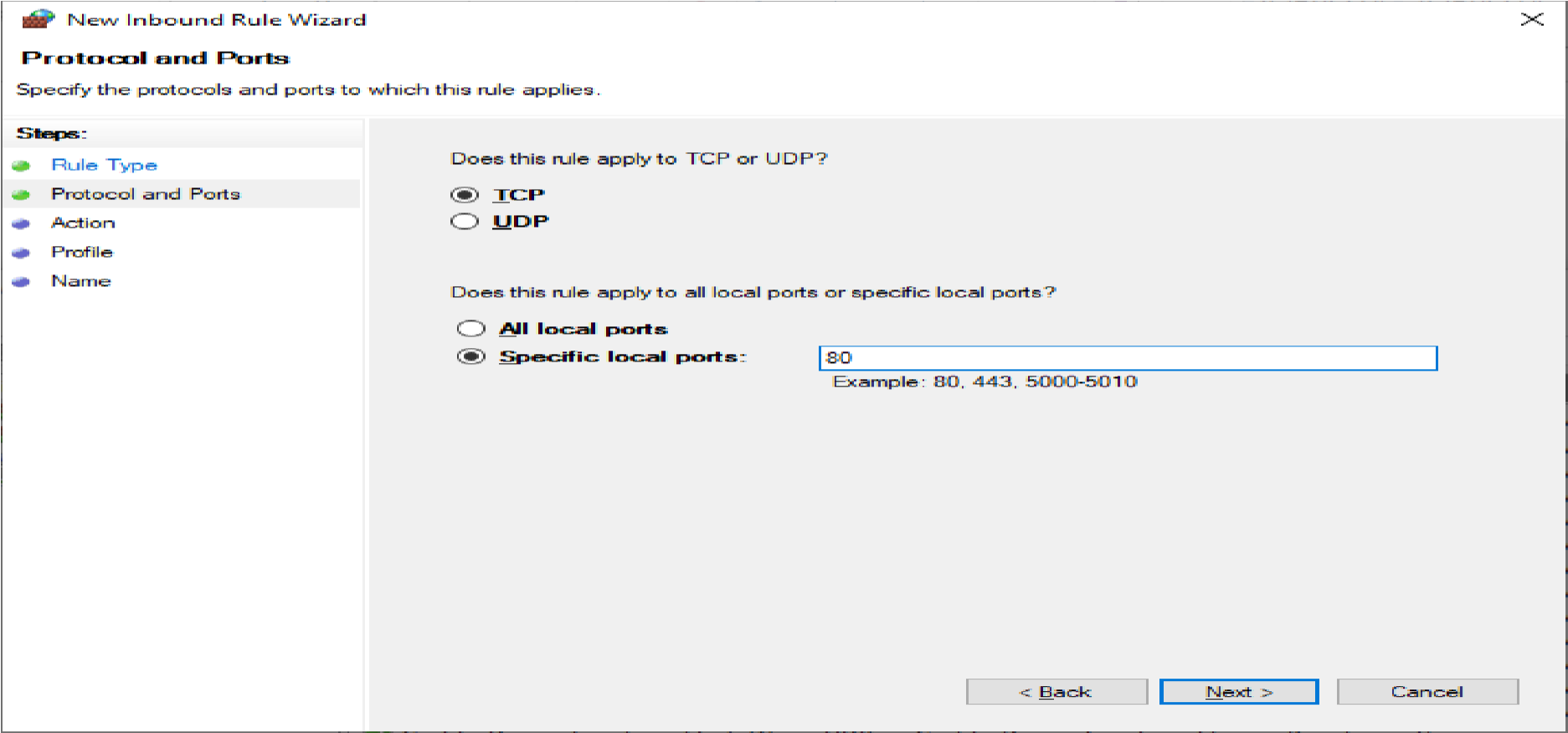
**2.Click On Inbound Rules.**



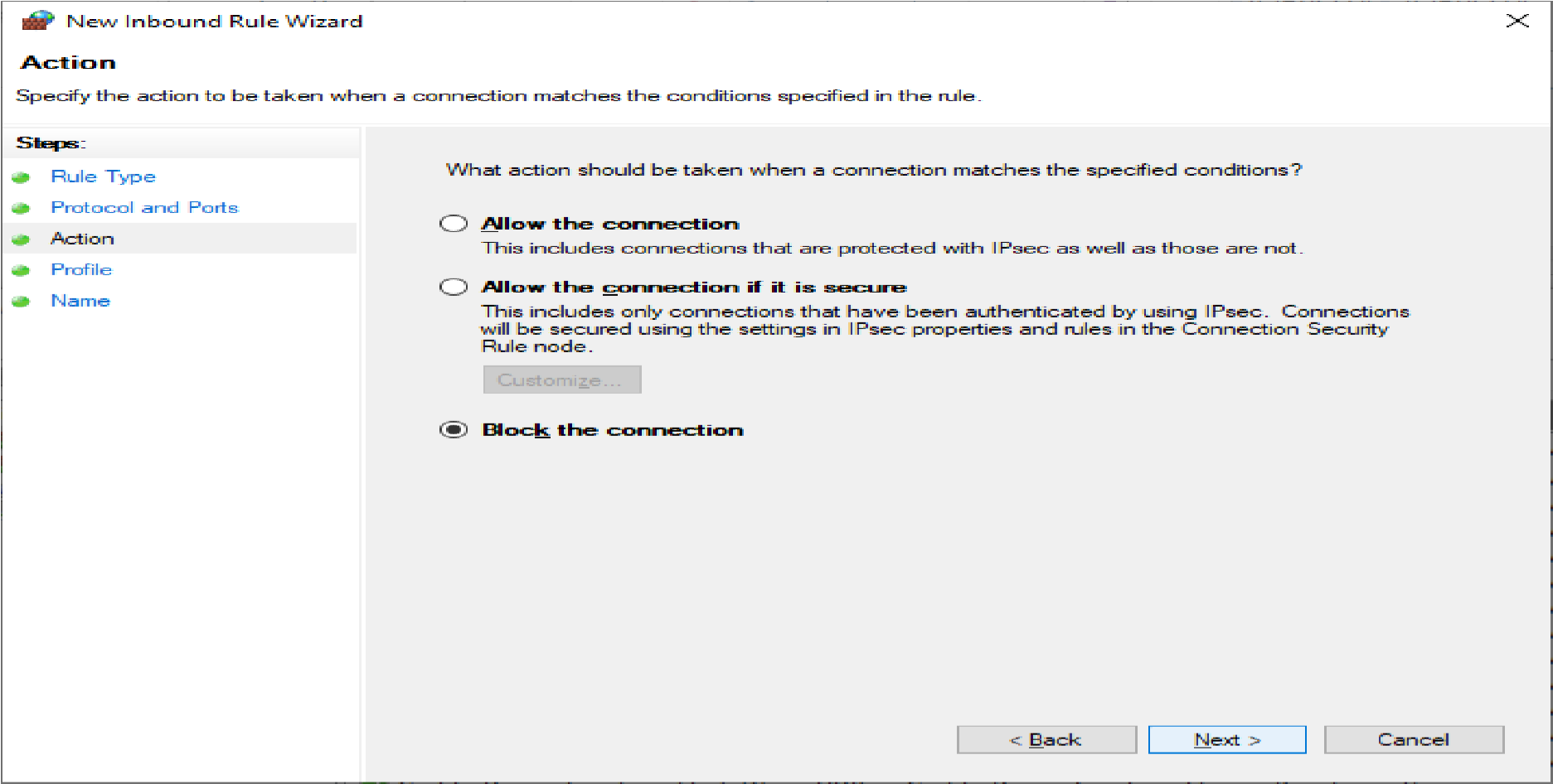
**3.Now Click on New Rule and click on port then click on next.**



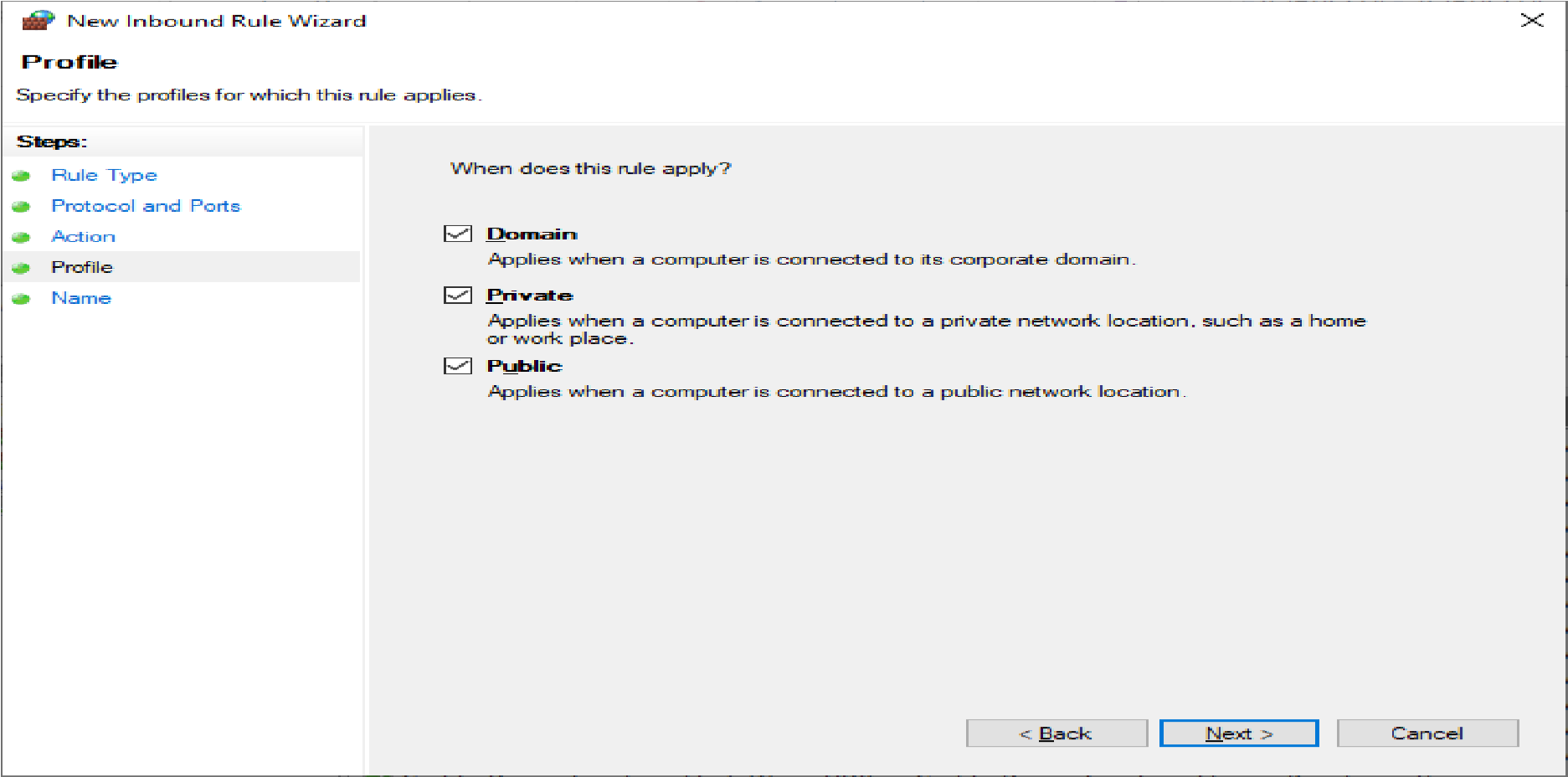
**4.Set the Local port and click on next.**



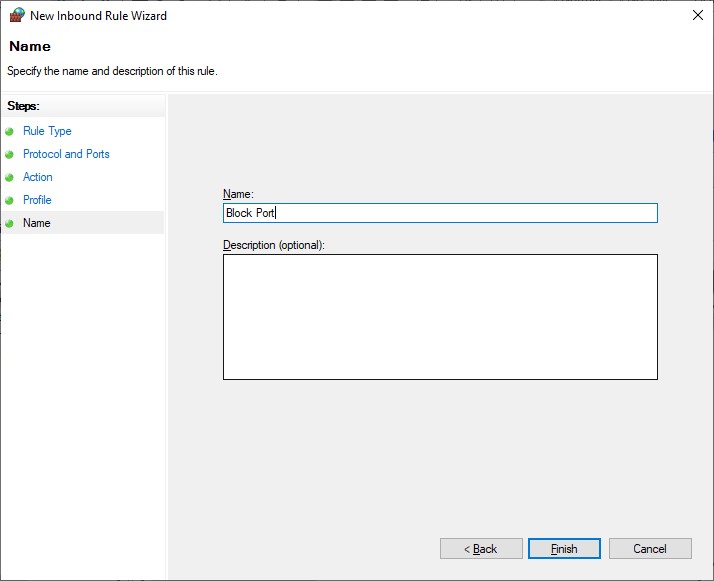
**5.Click on the Block the Connection then click on next.**



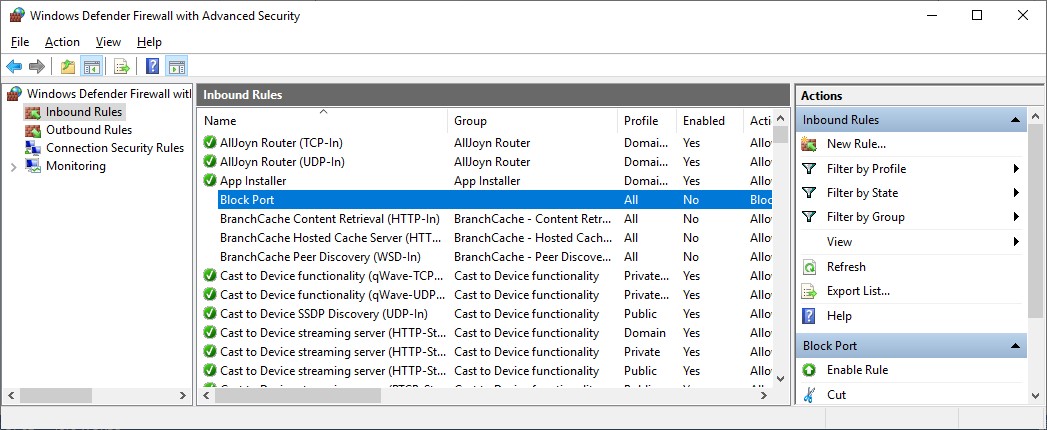
**6 .Click on the all three checkbox and click on next.**



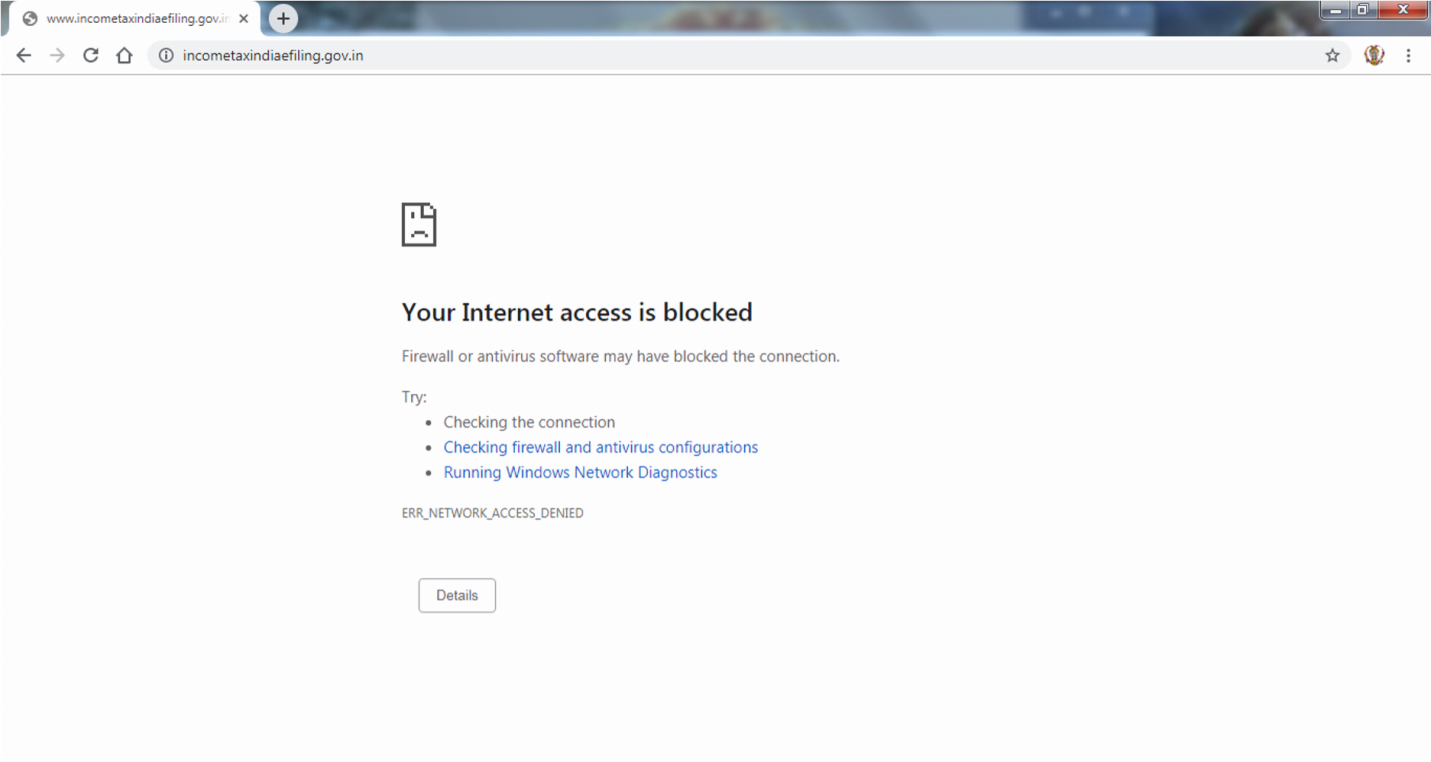
**7 .Set the rule name as Block Port and then finish it.**



**8 .Enable the BlockPort.**

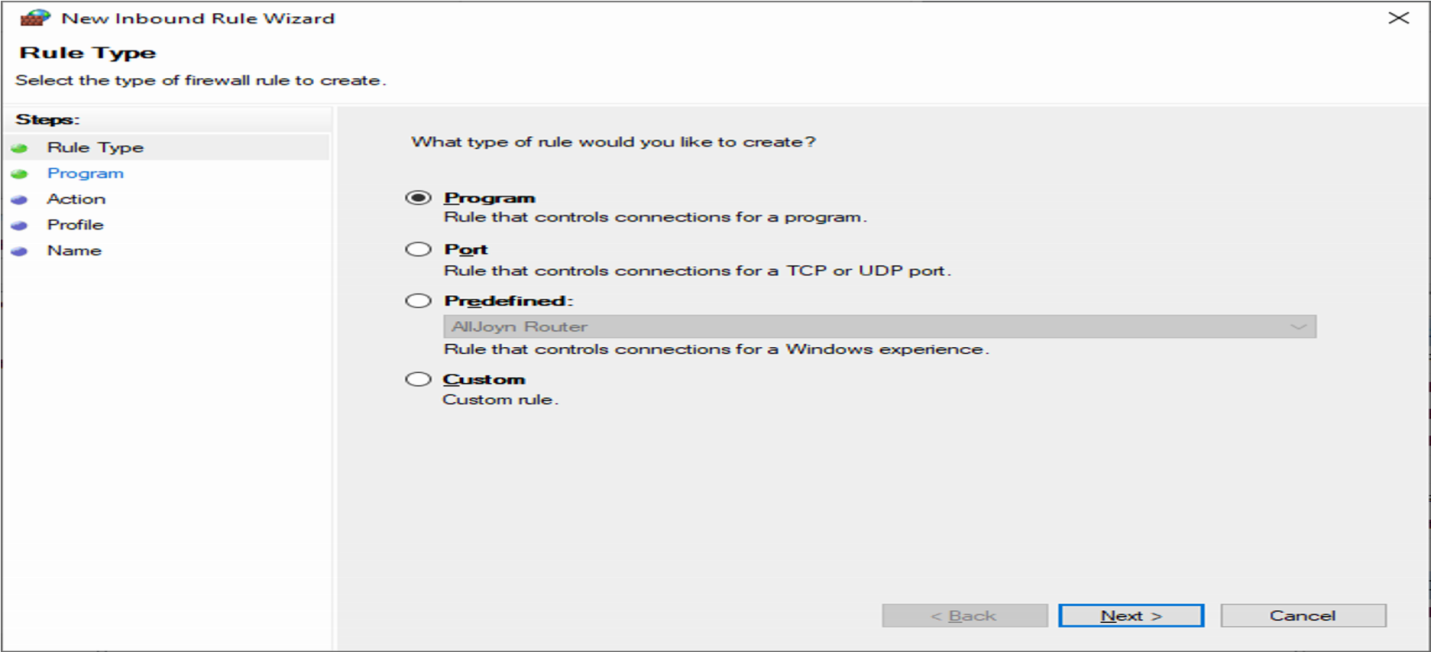


**9.Now Click on Outbound Rules and then follow the same steps as above in Inbound Rule and the Output is:**

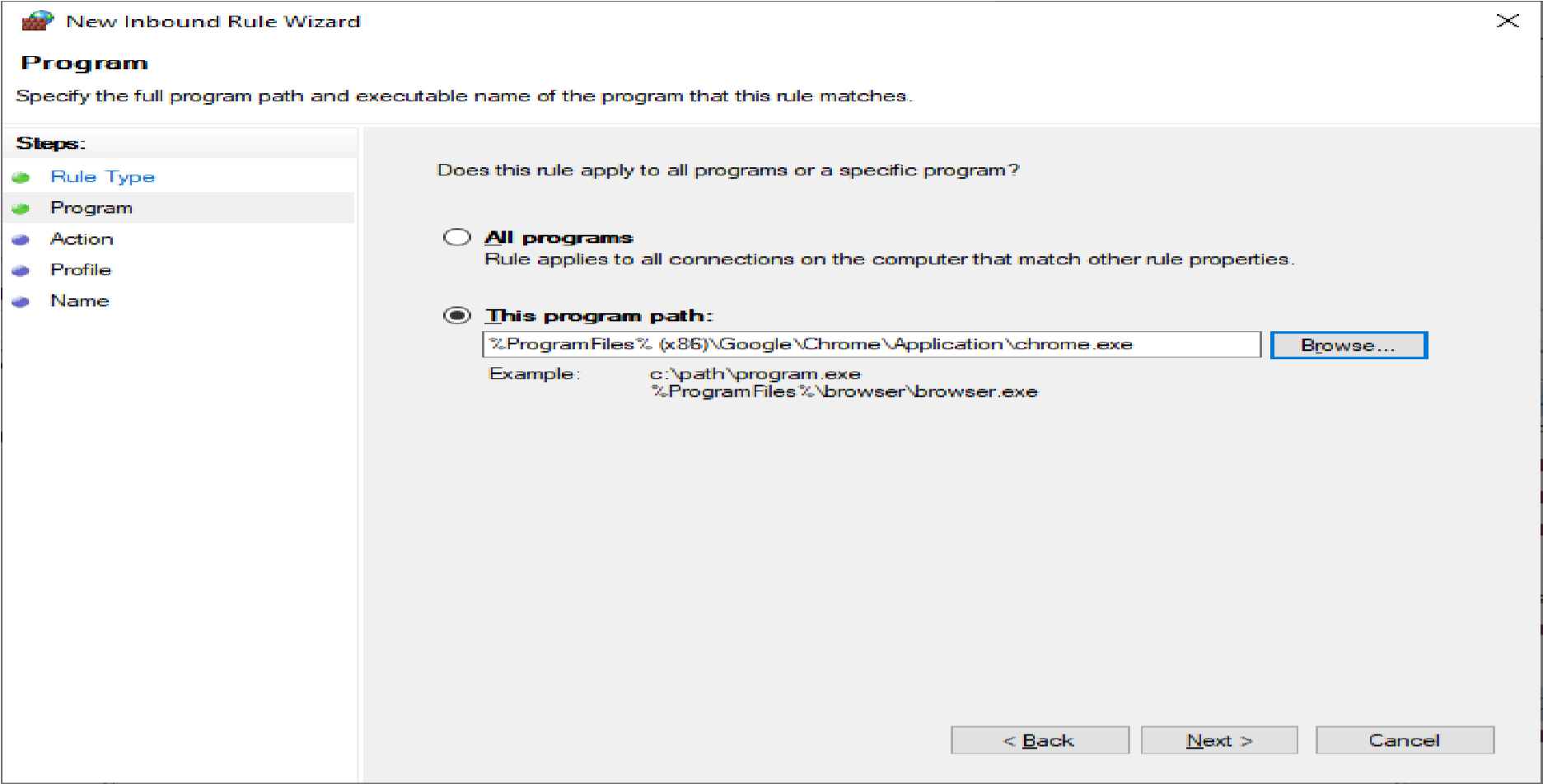


**For a Program:**

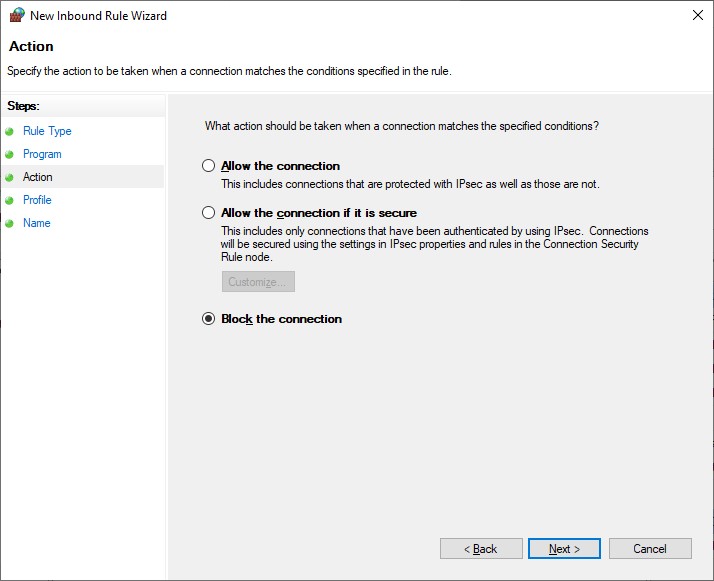
**1.In Windows Firewall and advanced setting Click Inbound Rule and then select New Rule then click on Program then next.**



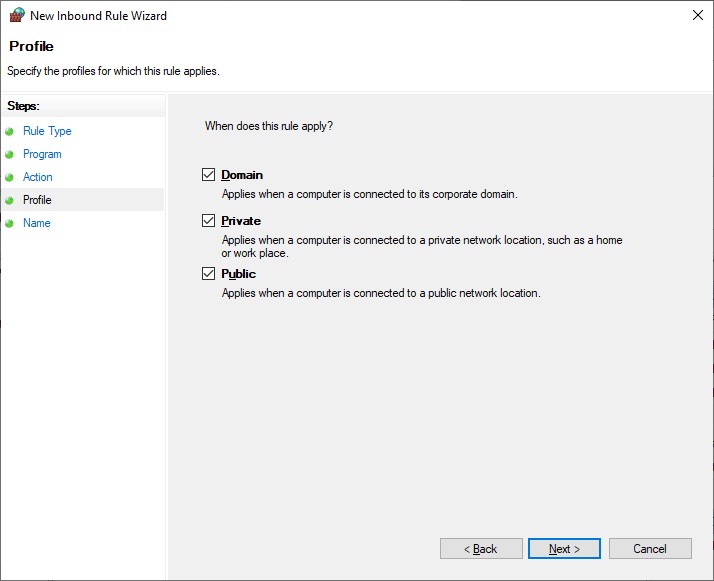
**2.Set the Path as you want or Browse it and then click on next.**



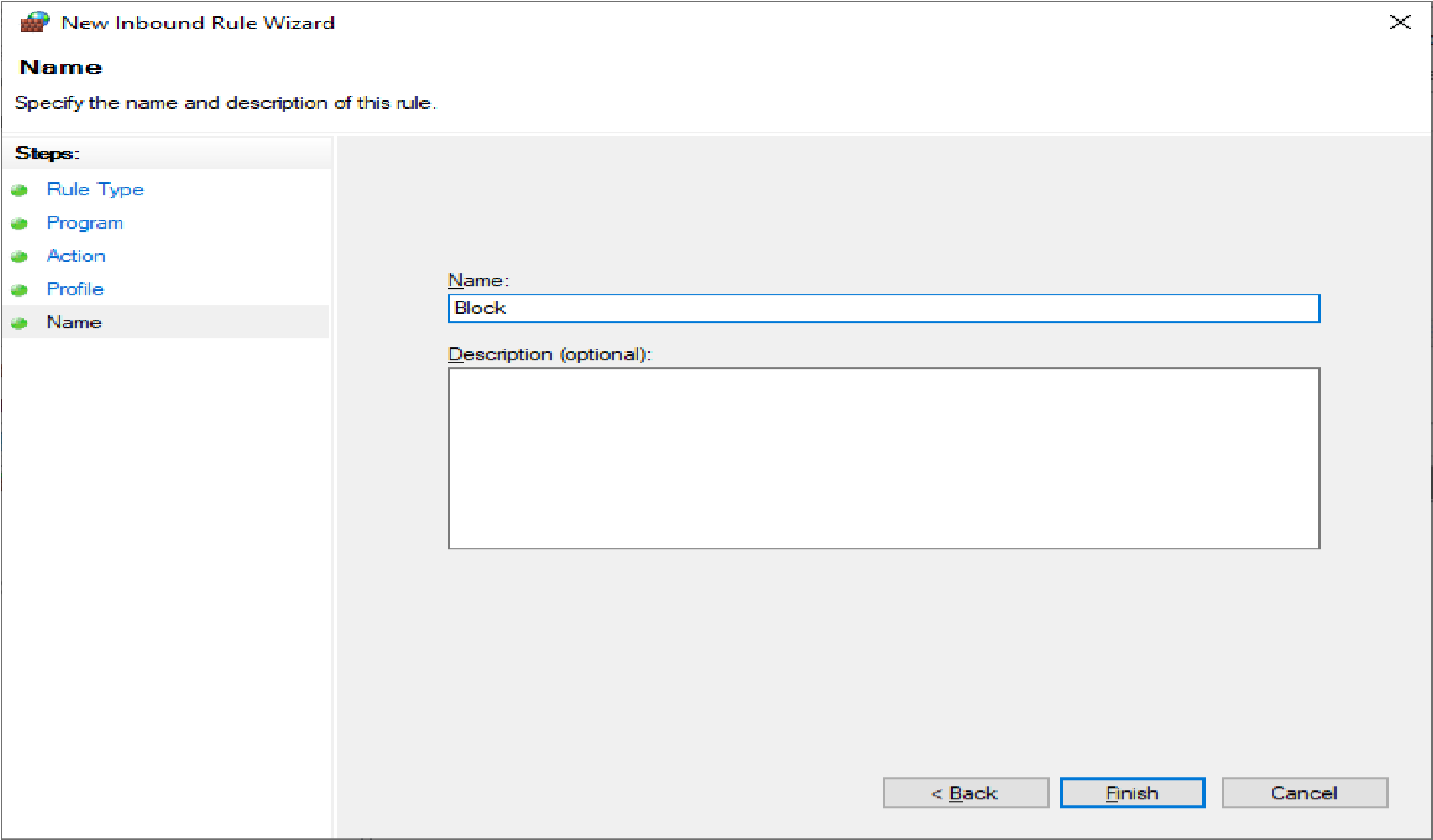
**3 .Block the Connection and click on next.**



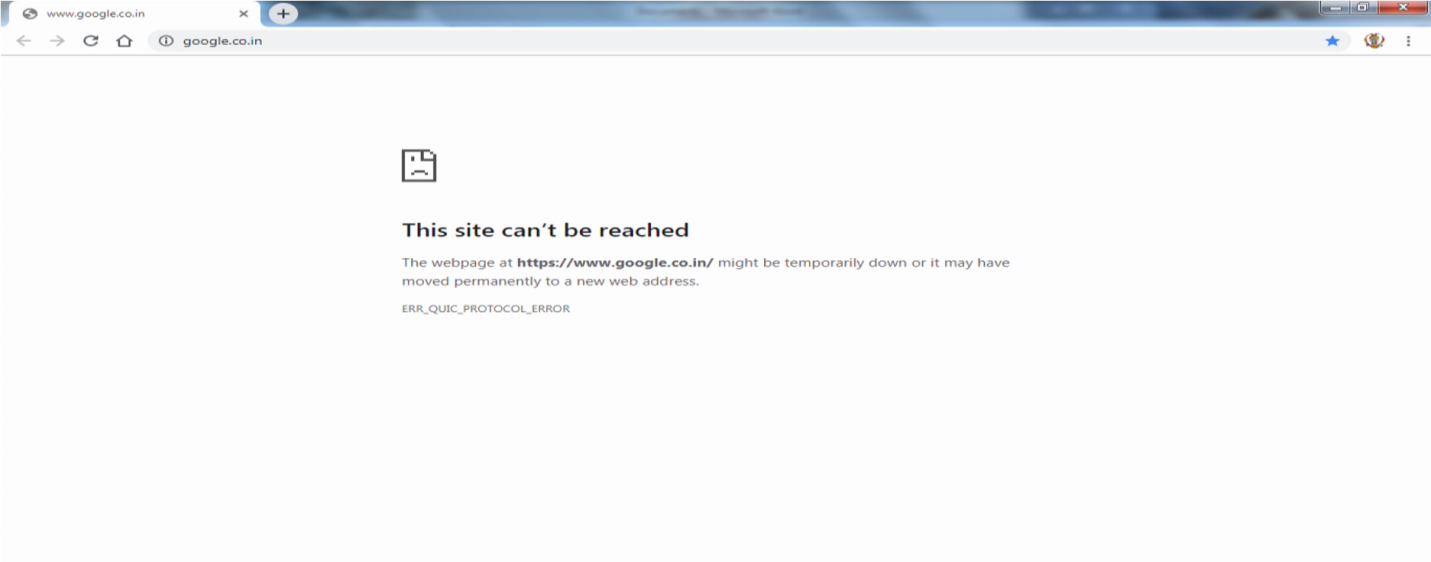
**4 .Click on all the checkbox and the next.**



**5 .Set the rule name as Block and then finish it.**

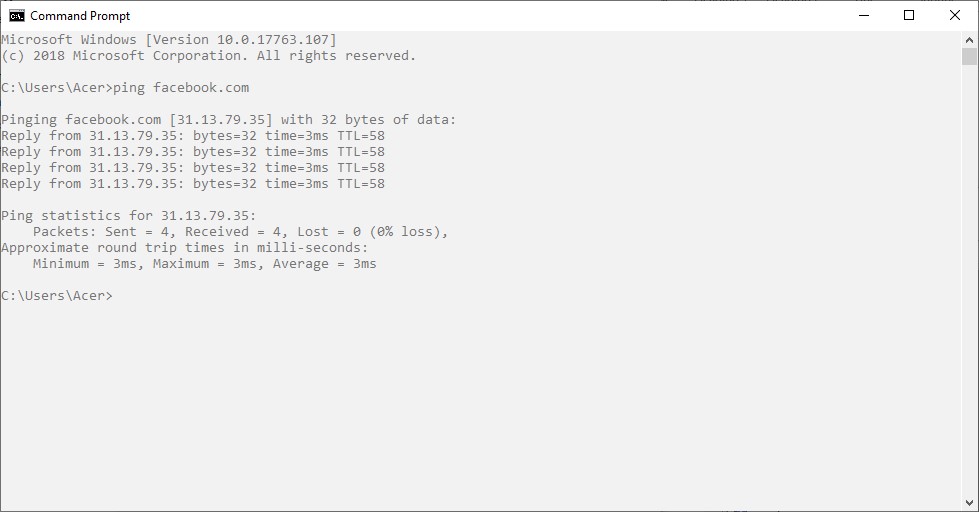


**6. Now Click on Outbound Rules and then follow the same steps as above in Inbound Rule and the Output is:**

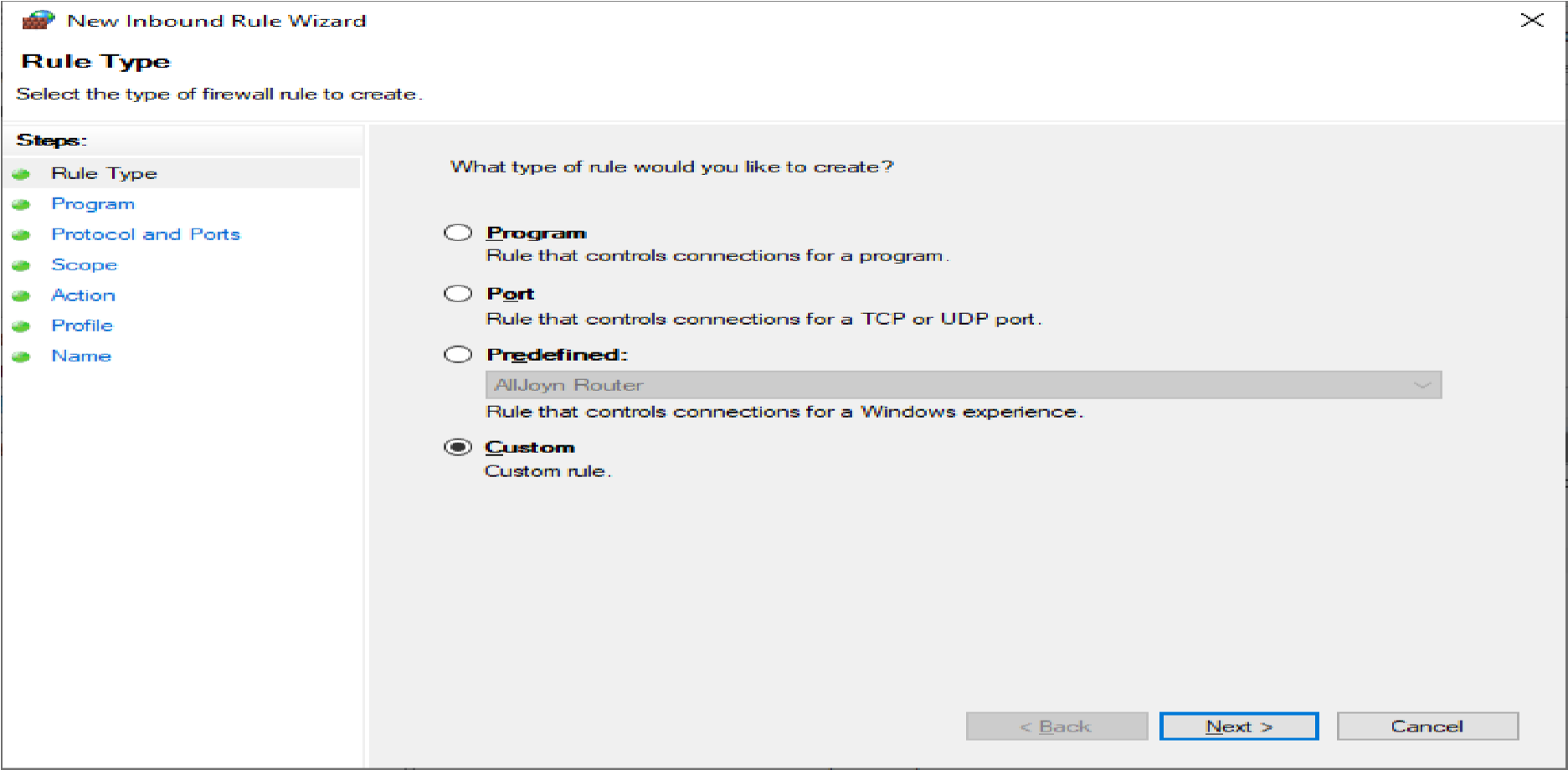


**For a Website:**

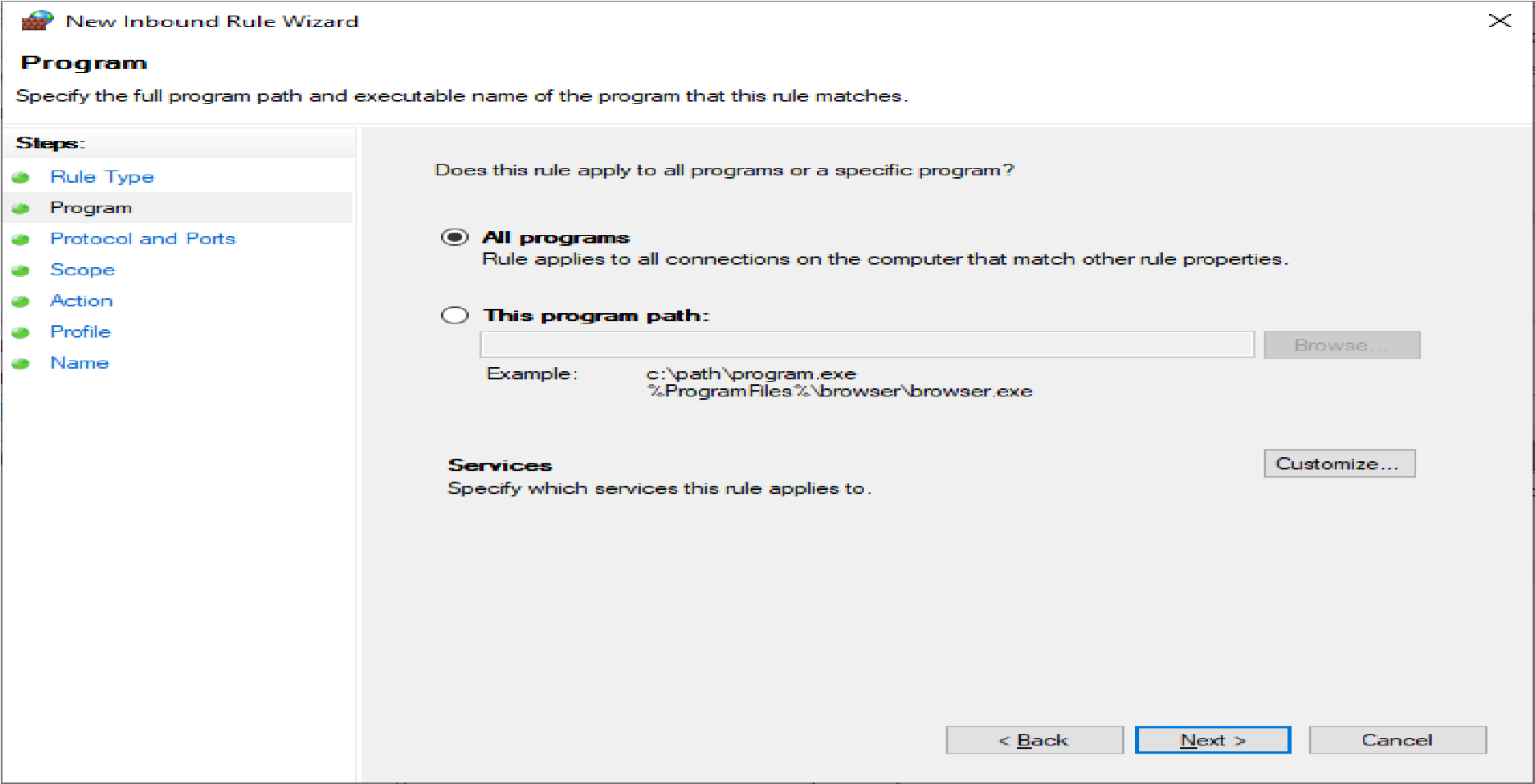
**1 .Find the IP address of any Website in cmd using ping command.**



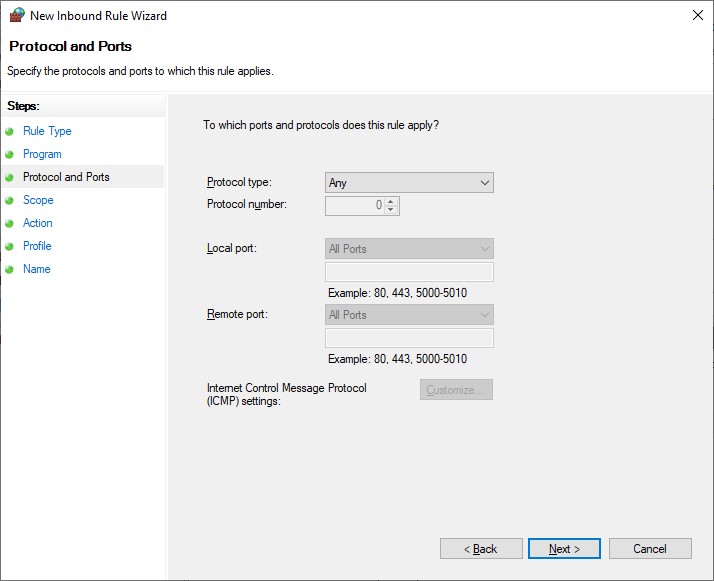
**2.In the Windows Firewall with Advanced Security Click on the Inbound Rule and select New Rule then Choose the Custom Rule type and click on next.**



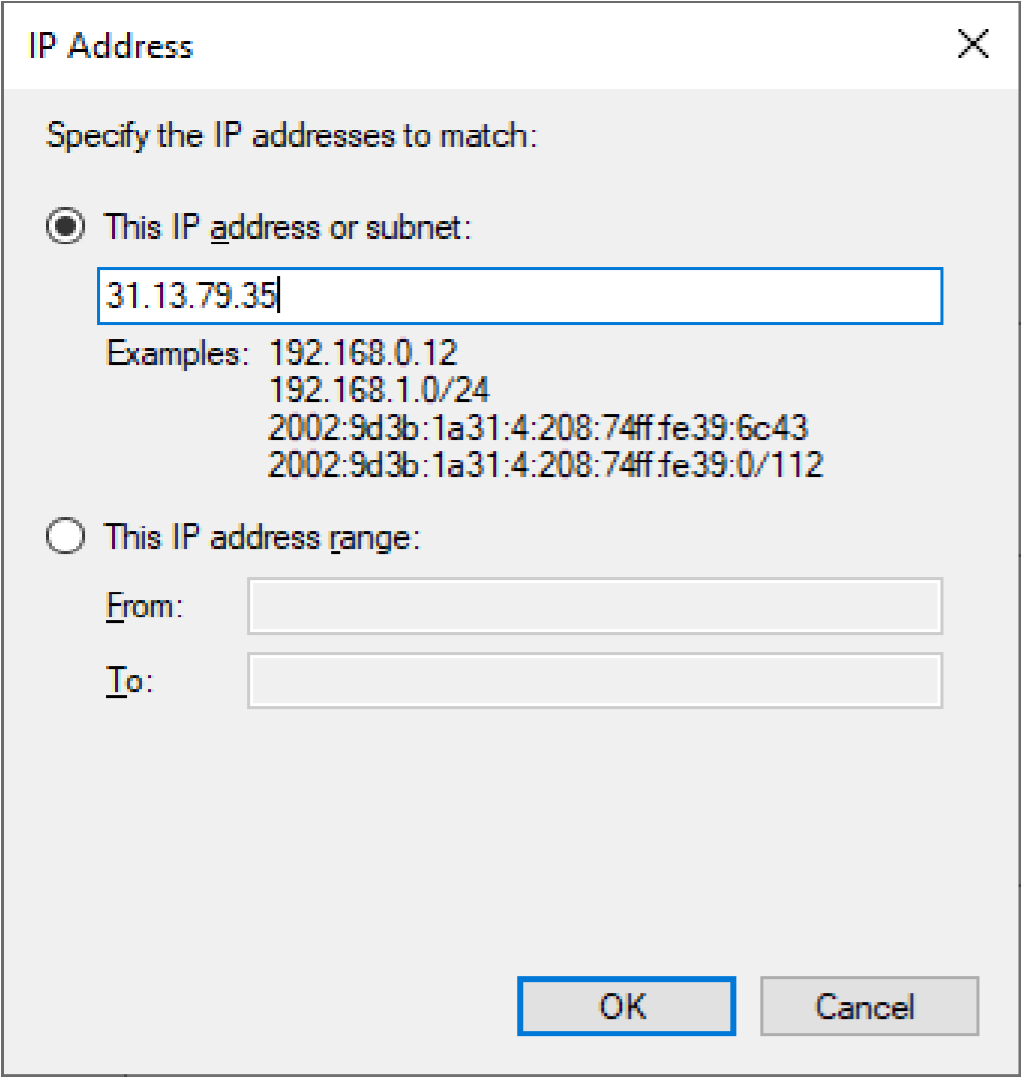
**3 .Click on Next.**



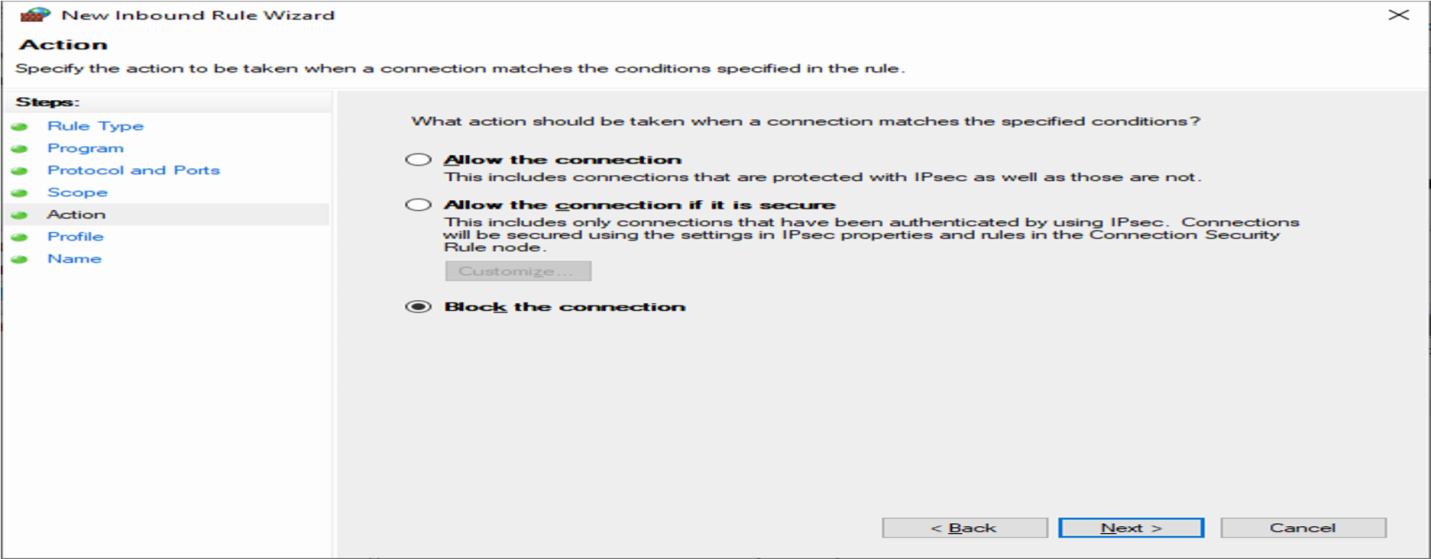
**4.Click on Next.**



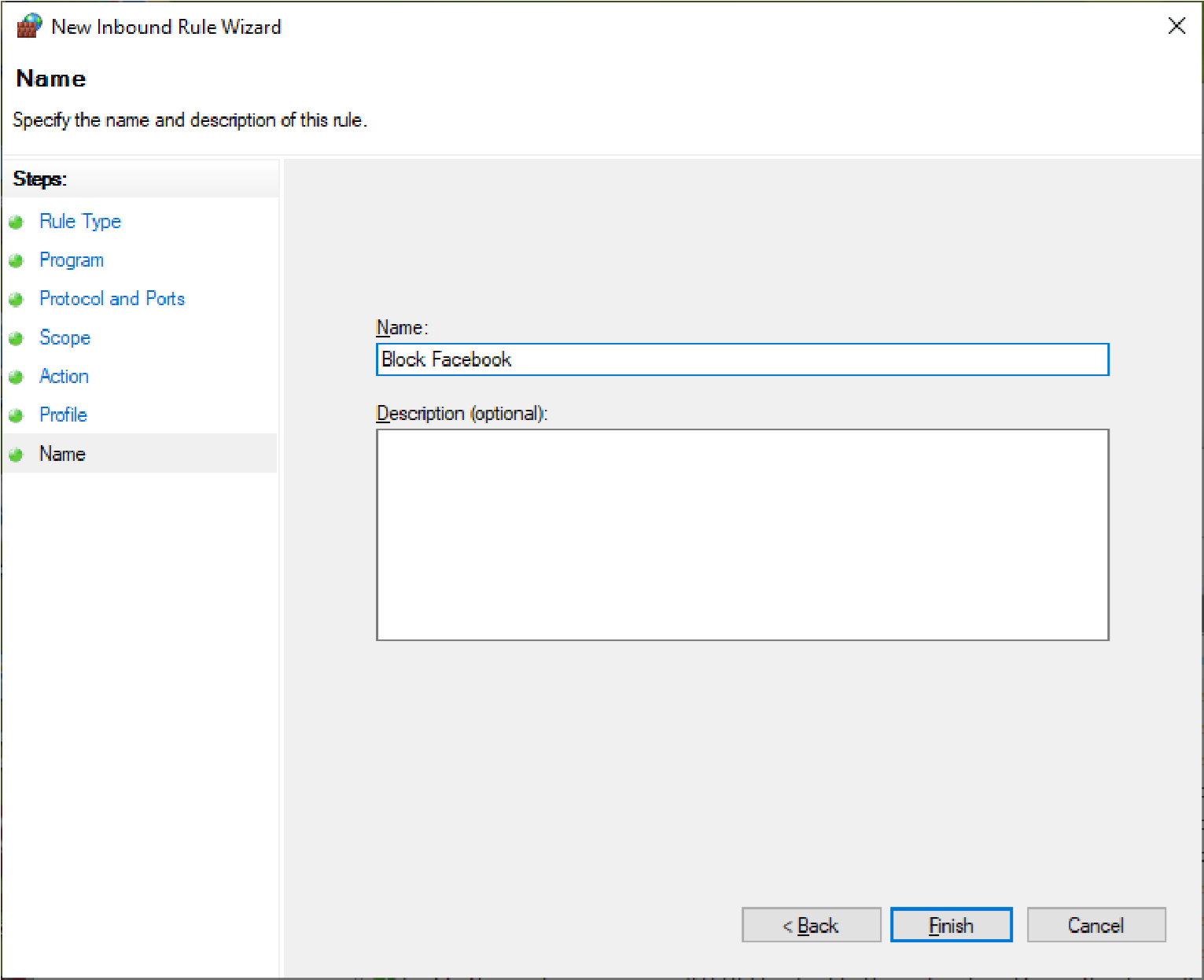
**5.Click on Next and Choose the These IP Address and add the IP Address which we want to Block and click on Ok.**



**6.Block the Connection and Press Next.**



**7.Again press next and Enter the Rule name as Block Facebook and Finish it.**



**8.Do the same steps for Outbound Rules and the Output is:**

