XOR WITH 0,1

```
#include<stdio.h>

#include<string.h>

void main()
{
   char ip[20]="HELLO WORLD",z,o;
   int i,len=strlen(ip);
   printf("XOR WITH 0: ");
   for(i=0;i<len;i++)
   {
      z=ip[i]^0;
      printf("%c",z);
   }
   printf("\nXOR WITH 1: ");
   for(i=0;i<len;i++)
   {
      o=ip[i]^1;
      printf("%c",o);
   }
}
```

OUTPUT:

XOR WITH 0: HELLO WORLD

XOR WITH 1: IDMMN!VNSME

AND OR XOR WITH 127

```
#include<stdio.h>
#include<string.h>
void main()
{
   char ip[20]="HELLO WORLD",a,o,x;
   int i,len=strlen(ip);
   printf("AND WITH 127: ");
   for(i=0;i<len;i++)
   {
      a=ip[i]&127;
      printf("%c",a);
   }
   printf("\nOR WITH 127: ");
   for(i=0;i<len;i++)
   {
      o=ip[i]|127;
      printf("%c",o);
   }
   printf("\nXOR WITH 127: ");
   for(i=0;i<len;i++)
   {
      x=ip[i]^127;
      printf("%c",x);
   }
}
```

OUTPUT:

AND WITH 127: HELLO WORLD
OR WITH 127: □□□□□□□□□□□
XOR WITH 127: 7:330_(0-3;

CAESAR CIPHER

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int main()
{
char m[100],c;
int i,k,val,op,b;
while(1)
{
printf("\n enter your choice:\n");
printf("1.encryption 2.decryption 3.exit\n");
scanf("%d",&op);
switch(op)
{
case 1:printf("enter the message: ");
    scanf("%s",m);
    printf("enter a key: ");
    scanf("%d",&k);
    for(i=0;m[i]!='\0';i++)
    {
    c=m[i];
    if(c>='a'&&c<='z')
    {
                c=c+k;
                if(c>'z')
                {
                c=c-'z'+'a'-1;
                }
                m[i]=c;
    }
    else if(c>='A'&&c<='Z')
    {
                c=c+k;
                if(c>'Z')
                {
                c=c-'Z'+'A'-1;
                }
                m[i]=c;
    }
    }
    printf("Encrypted message:%s\n",m);
    break;
```

```
case 2:printf("enter the message: ");
      scanf("%s",m);
      printf("enter a key: ");
      scanf("%d",&k);
      for(i=0;m[i]!='\0';i++)
      {
      c=m[i];
      if(c>='a'&&c<='z')
      {
                  c=c-k;
                  if(c<'a')
                  {
                  c=c+'z'-'a'+1;
                  }
                  m[i]=c;
      }
      else if(c>='A'&&c<='Z')
      {
                  c=c-k;
                  if(c<'A')
                  {
                  c=c+'Z'-'A'+1;
                  }
                  m[i]=c;
      }
      }
      printf("decrypted message:%s\n",m);
      break;
case 3:exit(1);
}
}
return 0;
}
```

OUTPUT:

```
 enter your choice:
1.encryption 2.decryption 3.exit
1
enter the message: hello
enter a key: 3
Encrypted message:khoor

 enter your choice:
1.encryption 2.decryption 3.exit
```

2
enter the message: khoor
enter a key: 3
decrypted message:hello

enter your choice:
1.encryption 2.decryption 3.exit
3

MONOALPHABETIC CIPHER

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
void main()
{
   char
pt[26]={'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'};
   char
ct[26]={'Q','W','E','R','T','Y','U','I','O','P','A','S','D','F','G','H','J','K','L','Z','X','C','V','B','N','M'};
   char inp[30],ci[30],pl[30];
   int i,j;
   printf("enter plain text:");
   scanf("%s",inp);
   for(i=0;i<strlen(inp);i++)
   {
      for(j=0;j<26;j++)
      {
         if(pt[j]==inp[i])
         {
            ci[i]=ct[j];
         }
      }
   }
   printf("\ncipher text(after encryptiom):%s\n",ci);
   for(i=0;i<strlen(ci);i++)
   {
      for(j=0;j<26;j++)
      {
         if(ci[i]==ct[j])
         {
            pl[i]=pt[j];
         }
      }
   }
   printf("plain text(after edecryption):%s\n",pl);
}
```

OUTPUT:

```
enter plain text:ABCDEF

cipher text(after encryptiom):QWERTY
plain text(after edecryption):ABCDEF
```

HILL CIPHER

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int mes[3][1],c[3][3],en[3][1],a[3][3];
void encryption();
void getkeymsg();
void main()
{
   getkeymsg();
   encryption();
}
void encryption()
{
   int i,j,k;
   for(i=0;i<3;i++)
   {
      for(j=0;j<1;j++)
      {
         for(k=0;k<3;k++)
         {
            en[i][j]=en[i][j]+a[i][k]*mes[k][j];
         }
      }
   }
   printf("\nencrpted string:");
   for(i=0;i<3;i++)
   {
      printf("%c",(char)((en[i][0]%26)+97));
   }
   printf("\n");
}
void getkeymsg()
{
   int i,j;
   char msg[3];
   printf("enter 3*3 matrix:\n");
   for(i=0;i<3;i++)
   {
      for(j=0;j<3;j++)
      {
         scanf("%d",&a[i][j]);
```

```
            c[i][j]=a[i][j];
        }
    }
    printf("enter a 3 letter string:\n");
    scanf("%s",msg);
    for(i=0;i<3;i++)
        mes[i][0]=msg[i]-97;
}
```

OUTPUT:

```
enter 3*3 matrix:
1 5 3
7 9 6
4 8 2
enter a 3 letter string:
hat

encrpted string:mho
```

VIGENER CIPHER

```
#include<stdio.h>
#include<string.h>
void main()
{
   char msg[50],key[50];
   int i,j;
   printf("enter plain text:");
   scanf("%s",msg);
   printf("enter key:");
   scanf("%s",key);
   int ml=strlen(msg),kl=strlen(key);
   char nk[ml],en[ml],de[ml];
   for(i=0,j=0;i<ml;i++,j++)
   {
      if(j==kl)
      j=0;
      nk[i]=key[j];
   }
   nk[i]='\0';
   for(i=0;i<ml;i++)
      en[i]=((msg[i]+nk[i])%26)+'A';
   en[i]='\0';
   for(i=0;i<ml;i++)
      de[i]=(((en[i]-nk[i])+26)%26)+'A';
   de[i]='\0';
   printf("\nOriginal message:%s",msg);
   printf("\nkey:%s",key);
   printf("\nNew generated key:%s",nk);
   printf("\nEncrypted message:%s",en);
   printf("\ndecrypted message:%s",de);
}
```

OUTPUT:

```
enter plain text:THE      CRAZY
enter key:HELLO

Original message:THECRAZY
key:HELLO
New generated key:HELLOHEL
Encrypted message:ALPNFHDJ
decrypted message:THECRAZY
```

VERNAME CIPHER

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int main()
{
   int i,p,k,ch,c;
   char pt[30],key[30],ct[30];
   while(1){
   printf("\n\n1.Encryption\n2.Decryption\n3.Exit\n");
   printf("Enter your choice:");
   scanf("%d",&ch);
   switch(ch)
   {
      case 1:printf("enter the plain text: ");
         scanf("%s",pt);
         p=strlen(pt);
         printf("enter the key: ");
         scanf("%s",key);
         k=strlen(key);
         if(p!=k)
         {
            printf("size of plaintext and key should be equal\n");
            exit(1);
         }
         else
         {
            printf("cipher text is: ");
            for(i=0;i<p;i++)
            {
               pt[i]=pt[i]-97;
               key[i]=key[i]-97;
               ct[i]=pt[i]^key[i];
               if(ct[i]>26)
               {
                  ct[i]-=26;
               }
               ct[i]+=97;
            }
         }
         printf("%s",ct);
         break;
      case 2:printf("enter the cipher text: ");
```

```
        scanf("%s",ct);
        c=strlen(ct);
        printf("enter the key: ");
        scanf("%s",key);
        k=strlen(key);
        if(c!=k)
        {
           printf("size of ciphertext and key should be equal\n");
           exit(1);
        }
        else
        {
           printf("plain text is: ");
           for(i=0;i<c;i++)
           {
              ct[i]=ct[i]-97;
              key[i]=key[i]-97;
              pt[i]=ct[i]^key[i];
              if(pt[i]>26)
              {
                 pt[i]-=26;
              }
              pt[i]+=97;
           }
        }
        puts(pt);
        break;
     case 3: exit(1);
     default:printf("Choose valid option");
   }
   }
   return 0;
}
```

OUTPUT:

1.Encryption
2.Decryption
3.Exit
Enter your choice:1
enter the plain text: oak
enter the key: son
cipher text is: coh

1.Encryption

2.Decryption
3.Exit
Enter your choice:2
enter the cipher text: coh
enter the key: sonsn
plain text is: qak

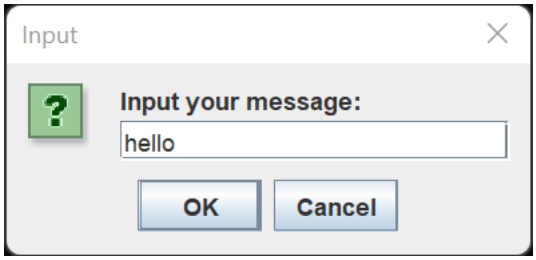
1.Encryption
2.Decryption
3.Exit
Enter your choice:3

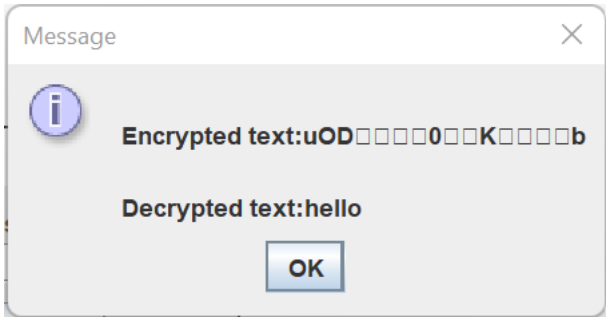
AES

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.swing.JOptionPane;
import javax.crypto.*;
import javax.crypto.spec.*;
import java.io.*;

public class aes
{
	public static void main(String[] args) throws Exception
	{
		KeyGenerator kgen=KeyGenerator.getInstance("AES");
		Cipher cipher=Cipher.getInstance("AES");
		SecretKey skey=kgen.generateKey();
		byte[] raw=skey.getEncoded();;
		SecretKeySpec skeyspec=new SecretKeySpec(raw,"AES");
		cipher.init(cipher.ENCRYPT_MODE,skey);
		String inputText=JOptionPane.showInputDialog("Input your message: ");
		byte[] encrypted=cipher.doFinal(inputText.getBytes());
		cipher.init(cipher.DECRYPT_MODE,skey);
		byte[] decrypted=cipher.doFinal(encrypted);
		JOptionPane.showMessageDialog(JOptionPane.getRootFrame(),"\nEncrypted
text:"+new String(encrypted)+"\n"+"\nDecrypted text:"+new String(decrypted));
	System.exit(0);
	}
}
```

OUTPUT:



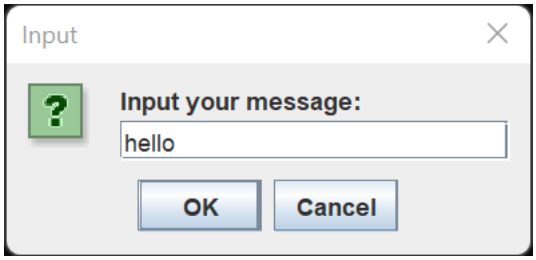

Message
Encrypted text:uOD□□□□0□□K□□□□b
Decrypted text:hello
OK

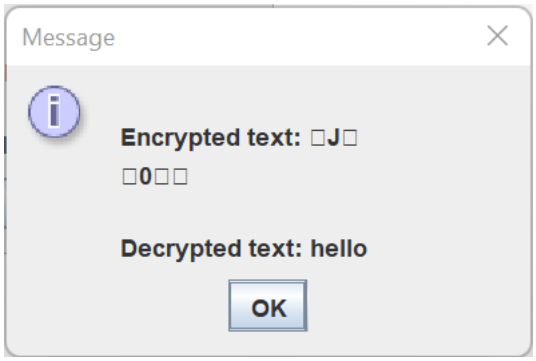
DES

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.swing.JOptionPane;
import javax.crypto.*;
import javax.crypto.spec.*;
import java.io.*;

public class des
{
	public static void main(String[] args) throws Exception
	{
		KeyGenerator kgen = KeyGenerator.getInstance("DES");
	   Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
	   SecretKey skey = kgen.generateKey();
		byte[] raw = skey.getEncoded();
	   SecretKeySpec skeySpec = new SecretKeySpec(raw, "DES");
	   cipher.init(Cipher.ENCRYPT_MODE,skey);
	   String inputText = JOptionPane.showInputDialog("Input your message: ");
	   byte[] encrypted = cipher.doFinal(inputText.getBytes());
		cipher.init(Cipher.DECRYPT_MODE,skey);
	   byte[] decrypted = cipher.doFinal(encrypted);
	   JOptionPane.showMessageDialog(JOptionPane.getRootFrame(), "\nEncrypted
text: " + new String(encrypted) + "\n" + "\nDecrypted text: " + new String(decrypted));
		System.exit(0);
	}
}
```

OUTPUT:



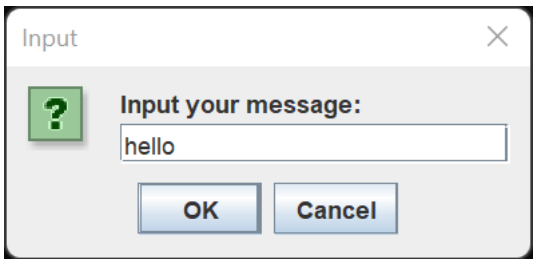

Message
Encrypted text: □J□
□0□□
Decrypted text: hello
OK

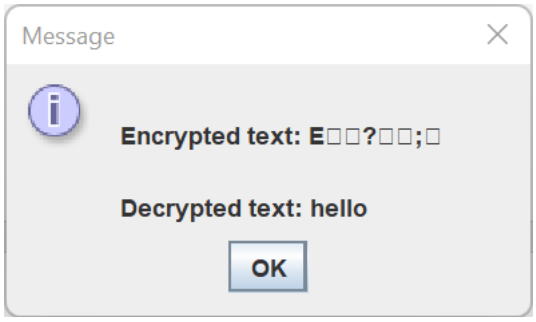
BLOWFISH

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.swing.JOptionPane;
import javax.crypto.*;
import javax.crypto.spec.*;
import java.io.*;

public class Blowfish
{
public static void main(String[] args) throws Exception {
  KeyGenerator kgen = KeyGenerator.getInstance("Blowfish");
  Cipher cipher = Cipher.getInstance("Blowfish");
  SecretKey skey = kgen.generateKey();
  byte[] raw = skey.getEncoded();
  SecretKeySpec skeySpec = new SecretKeySpec(raw, "Blowfish");
  cipher.init(Cipher.ENCRYPT_MODE,skey);
  String inputText = JOptionPane.showInputDialog("Input your message: ");
  byte[] encrypted = cipher.doFinal(inputText.getBytes());
  cipher.init(Cipher.DECRYPT_MODE,skey);
  byte[] decrypted = cipher.doFinal(encrypted);
  JOptionPane.showMessageDialog(JOptionPane.getRootFrame(), "\nEncrypted text: " + new
String(encrypted) + "\n" + "\nDecrypted text: " + new String(decrypted));
  System.exit(0);
 }
}
```

OUTPUT:




Message
Encrypted text: E□□?□□;□
Decrypted text: hello
OK

RC4

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int main()
{
   int k[4],p[4],s[8],t[8];
   int i,j,k1,t1;
   int temp=0;
   printf("enter k:\n");
   for(i=0;i<4;i++)
      scanf("%d",&k[i]);
   printf("enter p:\n");
   for(i=0;i<4;i++)
      scanf("%d",&p[i]);
   for(i=0;i<8;i++)
      s[i]=i;
   for(i=0;i<8;i++)
   {
      if(i<4)
         t[i]=k[i];
      else
         t[i]=k[i-4];
   }
   j=0;
   for(i=0;i<=7;i++)
   {
      j=(j+s[i]+t[i])%8;
      temp=s[i];
      s[i]=s[j];
      s[j]=temp;
   }
   printf("after ksa state vector is:\n");
   for(i=0;i<8;i++)
      printf("%d ",s[i]);
   printf("\nafter prga cipher text is:\n");
   i=0;
   j=0;
   while(i<4)
   {
      i=(i+1)%8;
      j=(j+s[i])%8;
      temp=s[i];
```

```
        s[i]=s[j];
        s[j]=temp;
        t1=(s[i]+s[j])%8;
        k1=s[t1];
        k1=k1^p[i-1];
        printf("%d ",k1);
    }
    return 0;
}
```

OUTPUT:

```
enter k:
1
2
3
6
enter p:
1
2
2
2
after ksa state vector is:
2 3 7 4 6 0 1 5
after prga cipher text is:
4 3 2 3
```

RSA

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

int findgcd(int n1,int n2)
{
   int i,gcd;
   for(i=1;i<=n1&&i<=n2;i++)
   {
      if(n1%i==0&&n2%i==0)
         gcd=i;
   }
   return gcd;
}

int power(int a,int b,int n)
{
   long long x=1,y=a;
   while(b>0)
   {
      if(b%2==1)
      x=(x*y)%n;
      y=(y*y)%n;
      b/=2;
   }
   return x%n;
}

int main()
{
   int p,q;
   int pi,m,n,e,d;
   printf("enter values for p and q:\n");
   scanf("%d",&p);
   scanf("%d",&q);
   n=p*q;
   pi=(p-1)*(q-1);
   for(e=5;e<=pi;e++)
   {
      if(findgcd(pi,e)==1)
         break;
   }
```

```
   for(d=e;d<=pi;d++)
   {
      if((d*e)%pi==1)
         break;
   }
   printf("enter the plain text M:\n");
   scanf("%d",&m);
   int ct=power(m,e,n);
   printf("encryption:\n");
   printf("%d\n",ct);
   int m1=power(ct,d,n);
   printf("decryption:\n");
   printf("%d\n",m1);
   return 0;
}
```

OUTPUT:

```
enter values for p and q:
17
11
enter the plain text M:
88
encryption:
11
decryption:
88
```

DIFFE HELLMAN

```
import java.io.*;
import java.util.Scanner;
public class DHKE
{
	public static void main(String[] args)
	{
		int q,alpha_picked,xa,xb,ya,yb,ka,kb,index=0;
		int alpha [] =new int[100];
		System.out.println("Enter the prime : ");
		Scanner sc=new Scanner(System.in);
		q=sc.nextInt();
		for(int i=2;i<q;i++)
		{
			int alpharnot[]=new int[q];
			for (int j=1;j<=q;j++)
			{
				alpharnot[j-1]=(int)((java.lang.Math.pow(i,j))%q);
				int c=0;
				for(int k=0;k<q;k++)
				{
					for(int p=k+1;p<q;p++)
					{
						if(alpharnot[k]==alpharnot[p])
						{
							c++;
						}
					}
				}
				if(c==0)
				{
					alpha[index]=i;
					index++;
				}
			}
		}
		for(int i=0;i<index;i++)
		{
			System.out.println("Primitive root is : "+ alpha[i]);
		}
		System.out.println("Select one of the root : ");
		alpha_picked=sc.nextInt();
		System.out.println("Select Xa: ");
```

```
                xa=sc.nextInt();
                System.out.println("Select Xb: ");
                xb=sc.nextInt();
                ya=(int)((java.lang.Math.pow(alpha_picked,xa))%q);
                yb=(int)((java.lang.Math.pow(alpha_picked,xb))%q);
                ka=(int)((java.lang.Math.pow(yb,xa))%q);
                kb=(int)((java.lang.Math.pow(ya,xb))%q);
                System.out.println("Ka: "+ka+" Kb :"+kb);
                if(ka==kb)
                {
                        System.out.println("Keys are same");
                }
        }
}
```

OUTPUT:

```
Enter the prime :
5
Primitive root is : 2
Primitive root is : 3
Select one of the root :
2
Select Xa:
2
Select Xb:
3
Ka: 4 Kb :4
Keys are same
```

SHA1

```
import java.security.*;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
public class SHA1
{
	public static void main (String [] a)
	{
	try
	{
		MessageDigest md = MessageDigest.getInstance("SHA1");
		System.out.println("Message digest object info: ");
		System.out.println(" Algorithm = " +md.getAlgorithm());
		System.out.println(" Provider = " +md.getProvider());
		System.out.println(" ToString = " +md.toString());
		String input = "";
		md.update(input.getBytes());
		byte[] output = md.digest();
		System.out.println("SHA1(\""+input+"\") = " +bytesToHex(output));
		input = "abc";
		md.update(input.getBytes());
		output = md.digest();
		System.out.println("SHA1(\""+input+"\") = " +bytesToHex(output));
		input = "abcdefghijklmnopqrstuvwxyz";
		md.update(input.getBytes());
		output = md.digest();
		System.out.println("SHA1(\"" +input+"\") = " +bytesToHex(output));
		System.out.println("");
	}
	catch (Exception e)
	{
		System.out.println("Exception: " +e);
	}
}
public static String bytesToHex(byte[] b)
{
	char hexDigit[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};
	StringBuffer buf = new StringBuffer();
	for (int j=0; j<b.length; j++)
	{
		buf.append(hexDigit[(b[j] >> 4) & 0x0f]);
		buf.append(hexDigit[b[j] & 0x0f]);
```

```
        }
        return buf.toString();
}
}
```

OUTPUT:

```
Message digest object info:
 Algorithm = SHA1
 Provider = SUN version 19
 ToString = SHA1 Message Digest from SUN, <initialized>

SHA1("") = DA39A3EE5E6B4B0D3255BFEF95601890AFD80709
SHA1("abc") = A9993E364706816ABA3E25717850C26C9CD0D89D
SHA1("abcdefghijklmnopqrstuvwxyz") = 32D10C7B8CF96570CA04CE37F2A19D84240D3A89
```

MD5

```
import java.security.*;
public class MD5
{
   public static void main(String[] a)
   {
      try
      {
         MessageDigest md = MessageDigest.getInstance("MD5");
         System.out.println("Message digest object info: ");
         System.out.println(" Algorithm = " +md.getAlgorithm());
         System.out.println(" Provider = " +md.getProvider());
         System.out.println(" ToString = " +md.toString());
         String input = "";
         md.update(input.getBytes());
         byte[] output = md.digest();
         System.out.println("MD5(\""+input+"\") = " +bytesToHex(output));
         input = "abc";
         md.update(input.getBytes());
         output = md.digest();
         System.out.println("MD5(\""+input+"\") = " +bytesToHex(output));
         input = "abcdefghijklmnopqrstuvwxyz";
         md.update(input.getBytes());
         output = md.digest();
         System.out.println("MD5(\"" +input+"\") = " +bytesToHex(output));
         System.out.println();
      }
      catch (Exception e)
      {
         System.out.println("Exception: " +e);
      }
   }
   public static String bytesToHex(byte[] b)
   {
      char hexDigit[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};
      StringBuffer buf = new StringBuffer();
      for (int j=0; j<b.length; j++)
      {
         buf.append(hexDigit[(b[j] >> 4) & 0x0f]);
         buf.append(hexDigit[b[j] & 0x0f]);
      }
      return buf.toString();
   }
```

}

OUTPUT:

```
Message digest object info:
 Algorithm = MD5
 Provider = SUN version 19
 ToString = MD5 Message Digest from SUN, <initialized>

MD5("") = D41D8CD98F00B204E9800998ECF8427E
MD5("abc") = 900150983CD24FB0D6963F7D28E17F72
MD5("abcdefghijklmnopqrstuvwxyz") = C3FCD3D76192E4007DFB496CCA67E13B
```