Data Security
Programs

ملاحظات عامة

1:- يتم طرح ٩٧ من كل حرف لتحويلهُ من ASCII إلى تسلسل.
ثم يتم تحويلهُ من تسلسل إلى ASCII بإضافة ٩٧ إلى الحرف.
٢:- يتم أعطاء قيمة فارغة لكل string عند التعريف.
٣:- تكون عملية فك الشفرة عكس عملية التشفير أي أن الجمع يصبح طرح والضرب قسمة وهكذا.
٤:- يتم التعامل مع النص حرف تلوا الأخر عن طريق for loop بعدد أحرف ذلك النص أو foreach.
٥:- الغاية من استعمال الأقواس في العمليات الحسابية هي لزيادة الأسبقية.
٦:- جميع البرامج Console ويمكنكم طلب نسخة برامج Windows

البرنامج الرئيسي
يكون البرنامج الرئيسي عادةً نفسهُ لأغلب خوارزميات وطرق التشفير، ويكتب كالتالي

```
static void Main(string[] args)
{
   Console.Write("enter the Key: ");
   int k=int.Parse(Console.ReadLine());
   Console.WriteLine("Enter E for Encryption\nEnter D for Decryption");
   int ch =Console.ReadKey(true).KeyChar;
   if(ch == 'E' || ch == 'e')
   {
      Console.Write("enter plain text: ");
      string st=Console.ReadLine();
      Enc(st,k);
   }
   else if(ch == 'D' || ch == 'd')
   {
      Console.Write("enter cipher text: ");
      string st=Console.ReadLine();
      Dec (st, k);
   }
   Console.ReadKey();
}
```

طرق التشفير

سوف نتطرق لمجموعة من طرق التشفير وكتابة كل من دالة التشفير ودالة فك الشفرة لكل من تلك الطرق

1- Column Transposition

```
static string KeyModefire(string Key1)
{
   char[] tkey = Key1.ToArray<char>();
   for (int i = 0; i < tkey.Length; i++)
      for (int j = i + 1; j < tkey.Length; j++)
         if (tkey[i] > tkey[j])
         {
            char t = tkey[i];
            tkey[i] = tkey[j];
            tkey[j] = t;
         }
   string key2 = "";
   string key3 = "";
   for (int i = 0; i < Key1.Length; i++)
      key2 += tkey[i];
   for (int i = 0; i < Key1.Length; i++)
      key3 += key2.IndexOf(Key1[i]);
   return (key3);
}
```

```
static void Enc(string p,string Key)
{
        int row = (int) Math.Ceiling(p.Length / (float)Key.Length);
        int column=Key.Length;
        char[,] A1=new char[row,column];
        char[,] A2=new char[row,column];
        int i = 0;
   for (int x = 0; x < column; x++)
      for (int y = 0; y < row; y++)
      {
         if(i >= p.Length)
            A1[y,x] = 'x';
         else
            A1[y,x] = p[i++];
      }
      for (i = 0; i < column; i++)
         for (int j = 0; j < row; j++)
            A2[j,i] = A1[j,Key[i]-48];
      for (i = 0; i < row; i++)
         for (int j = 0; j < column; j++)
            Console.Write(A2[i,j]);
}

static void Dec (string c, string Key)
{
   int column = Key.Length;
   int row = (int)Math.Ceiling(c.Length / (float)column);
   char[,] A1=new char[row,column];
   char[,] A2 = new char[row, column];
   int i = 0;
   for (int x = 0; x < row; x++)
      for (int y = 0; y < column; y++)
         A1[x,y] = c[i++];
   for (i = 0; i < column; i++)
      for (int j = 0; j < row; j++)
         A2[j,Key[i]-48] = A1[j,i];
   for(i = 0; i < column; i++)
      for (int j = 0; j < row; j++)
         Console.Write(A2[j,i]);
}
```

2- Addition Cipher (Direct Standard Or Caesar Cipher)

Encryption Formula C = (M + K) Mod 26

Decryption Formula M = (C – K)

```
static void Enc(string p, int k)
 {
    string c="";          //cipher text string
    foreach(char ch in p)
       c += Convert.ToChar((ch - 97 + k ) % 26 + 97);
                                                    // C = ( M + K ) Mod 26
    Console.Write("the cipher Text are: "+c);
 }

 static void Dec (string c, int k)
 {
    string p="";          //plain text string
    foreach(char ch in c)
    {
       int n;
       if (( n= ch - 97 - k ) < 0)       // M = ( C – K )
          n+=26;
       p += Convert.ToChar(n+97);
    }
    Console.Write("the Plan Text are: "+p);
 }
```

3- Zigzag Transposition Cipher

```
 static void Enc(string p)
{
   string c1 = "",c2="";
        for(int i=0;i<p.Length;i++)
        {
                c1 += p[i];
      if(++i<p.Length)
              c2 += p[i];
        }
   string ci = c1 + c2;
        Console.WriteLine("The Cipher Text is: " + ci);
}

static void Dec(string ci)
{
        string p = "";
   int n = (int)Math.Ceiling(ci.Length / 2.0);
   for (int x = 0; x < n; x++)
   {
      p += ci[x];
      if(x + n<ci.Length)
        p += ci[x + n];
   }
   Console.WriteLine("The Plan Text is: "+p);
}
```

4- Multiplicative Cipher

Encryption Formula C = (M * K) Mod 26

Decryption Formula M = (C / K)

```
static void Enc (string p,  int k)
 {
    string c="";
    foreach(char ch in p)
       c += Convert.ToChar((( ch - 97) * k) % 26 + 97);
    Console.Write("the cipher Text are: "+c);
 }

 static void Dec (string c, int k)
 {
    string p="";
    foreach (char ch in c)
    {
       int n = ch - 97;
       while (n % k != 0)
          n += 26;
       p += Convert.ToChar(n / k + 97);
    }
    Console.WriteLine("Plain Text is: "+p);
 }
```

5- Standard Reverse Cipher

Encryption Formula C = (25 – M) Mod 26

Decryption Formula M = (25 – C) Mod 26

كما نلاحظ فإن هذه الطريقة تستخدم نفس القانون للتشفير ولفك الشفرة

```
static void revers(string p)
{
   string c="";
   foreach(char ch in p)
   c += Convert.ToChar((25 - (ch - 97 )) % 26 + 97);
   Console.WriteLine("the cipher Text are: "+c);
}

static void Main()
{
   Console.Write("enter The text to Enc or Dec: ");
   string st=Console.ReadLine();
   revers (st);
   Console.ReadKey();
}
```

6- Affine Cipher

Encryption Formula C = (M * K1 + K2) Mod 26

Decryption Formula M = ((C – K2) / K1) Mod 26

من شروط هذه الطريقة أن القاسم المشترك الأعظم بين K1 و ٢٦ يساوي واحد، لذا نحتاج إلى دالة ليجاد القاسم المشترك الأعظم.

```
        دالة ليجاد العامل المشترك الأعظم //
static int GCD(int x, int y)
{
   if (y == 0)
      return (x);
   return (GCD(y, x % y));
}
```

طريقة ثانية

```
static int GCD(int x, int y)
{
   while (y != 0)
   {
      int z = x % y;
      x = y;
      y = z;
   }
   return (x);
}

static void Enc(string p, int k1, int k2)
{
   if( GCD ( k1, 26) == 1)
   {
      string c="";
      foreach(char ch in p)
         c += Convert.ToChar((( ch - 97 ) * k1 + k2 ) % 26 + 97);
         Console.WriteLine("the cipher Text are: "+c);
   }
   else
      Console.WriteLine(" unvalued key");
}
```

```
static void Dec(string c, int k1, int k2)
{
   if (GCD(k1, 26) == 1)
   {
      string p = "";
      foreach (char ch in c)
      {
         int n;
         if ((n = ch - 97 - k2) < 0)
            n += 26;
         while (n % k1 != 0)
            n += 26;
         p += Convert.ToChar(n / k1 + 97);
      }
      Console.WriteLine("the Plain Text are: " + p);
   }
   else
      Console.WriteLine("unvalued Key");
}
```

7- Keyword Mixed

```
static string mixed_array (string key1, char kl)
{
   string key = "";
   int x = 0;
   foreach(char ch in key1)
      if (!key.Contains(ch))
         key += ch;
   char[] mix=new char[27];
   char a = 'a';
   x = 0;
   int i;
   for (i = kl - 97; x < key.Length; i++)
      mix[i]= key[x++];
   for (; i < 26; i++)
   {
      if(!key.Contains(a))
         mix[i] = a;
      else i--;
      a++;
   }
   for( i = 0; i < kl - 97; i++)
   {
      if (!key.Contains(a))
         mix[i] = a;
      else i--;
      a++;
   }
   string m = "";
   for (i = 0; i < mix.Length; i++)
      m += mix[i];
   return m;
}
```

```
static void Enc ( string p, string key, char k)
{
   string c="";
   string mix = mixed_array ( key, k);
   foreach(char ch in p)
      c += mix[ch - 97];
   Console.WriteLine("The Cipher Text is: "+c);
}

static void Dec(string c,string key,char k)
{
   string p="";
   string  mix = mixed_array ( key, k);
   foreach(char ch in c)
      for (int j = 0; j < 26; j++)
         if (mix[j] == ch)
         {
            p += Convert.ToChar(j + 97);
            break;
         }
   Console.WriteLine("The Plain Text is: "+p );
}
```

8- Vigenere Cipher

Encryption Formula C = (M + K) % 26

Decryption Formula M = (C – K)

```
static void Enc (string p, string k)
{
   string  c= "\0";
   int x = 0;
   foreach(char ch in p)
   {
      x = ( x == k.Length)? 0 : x;
      c += Convert.ToChar((( ch - 97) + (k[x++] - 97)) % 26 + 97);
   }
   Console.WriteLine("The Cipher Text is: "+c);
}
```

```
static void Dec (string c, string k)
{
   string p="";
   int x=0;
   foreach(char ch in c)
   {
      int n;
      x = ( x == k.Length)? 0 : x;
      n = (( ch - 97) -  (k[x++] - 97));
      p += Convert.ToChar((n < 0)? n + 26 + 97 : n + 97);
   }
   Console.WriteLine("The Plain Text is: "+p);
}
```

9- Beaufort Cipher

Encryption Formula C = (K – M)

Decryption Formula M = (K – C)

```
static void Enc (string p, string k)
{
        string c = "";
        int x = 0;
   foreach (char ch in p)
   {
      int n;
      x = (x == k.Length) ? 0 : x;
      n = ((k[x++] - 97) - (ch - 97));
      if (n < 0)
         n += 26;
      c += Convert.ToChar(n + 97);
   }
   Console.WriteLine("Cipher Text is: "+c);
}
```

```
static void Main ( )
{
   Console.Write("enter the keyword: ");
   string k = Console.ReadLine();
   Console.WriteLine("Enter E for Encrption\nEnter D for Decrption\n");
   int c = Console.ReadKey(true).KeyChar;
   if(c == 'E' || c == 'e')
      Console.Write("enter plinte text: ");
   else if(c == 'D' || c == 'd')
      Console.Write("enter cipher text: ");
   string p = Console.ReadLine();
   Enc (p, k);
   Console.ReadKey();
}
```

10- LFSR(Linear Feedback Shift Register)

```
static string MaxSeq(string t1,string t2)
{
   string L1 = "";
   int n = t2.Length;
   int[] iv = new int[n];
   string s = "";
   for (int i = 0; i < n; i++)
   {
      iv[n - 1 - i] = t1[i] - 48;
      if (t2[n - i - 1] == '1')
         s += i;
   }
   int p = (int)Math.Pow(2, n) - 1;
   int[] maxs = new int[p];
   for (int i = p - 1; i >= 0; i--)
   {
      maxs[i] = iv[n - 1];
      int m = iv[s[0] - 48];
      for (int j = 1; j < s.Length; j++)
         m = m ^ (iv[s[j] - 48]);
      for (int j = n - 1; j > 0; j--)
         iv[j] = iv[j - 1];
      iv[0] = m;
      L1 += maxs[i];
```

```
    }
    return(L1);
}

static void Enc(string t3,string L1)
{
    string t4 = "";
    string s = t3;
    string k = L1;
    string binary = "";
    for (int i = 0; i < s.Length; i++)
    {
        int m = (int)s[i];
        for (int j = 0; j < 8; j++)
        {
            binary = m % 2 + binary;
            m /= 2;
        }
    }
    int x = 0;
    for (int i = 0; i < binary.Length; i++)
    {
        t4 += binary[i] ^ k[x];
        x = (x + 1) % k.Length;
    }
    Console.WriteLine(t4);
}

static void Main(string[] args)
{
    Console.Write("enter thr initial value: ");
    string t1 = Console.ReadLine();
    Console.Write("enter thr Cofitiont value: ");
    string t2 = Console.ReadLine();
    string L1 = MaxSeq(t1, t2);
    Console.WriteLine(L1);
    string t3 = Console.ReadLine();
    Enc(t3, L1);
    Console.ReadKey();
}
```

حساب قيمة Euler

```
static bool Prime (int N)
{
        for (int i = 2; i < N / 2; i++)
                if (N % i == 0)
                        return false;
        return true;
}

static long Euler (int N)
{
        if (Prime (N))
                return (N - 1);
        int m = 2, k = 2;
        while (Math.Pow(k, m) <= N)
        {
                while (Math.Pow(k,m) <= N)
                {
                        if ((Math.Pow(k, m) == N) && (Prime (k) ))
                                return (long)(Math.Pow(k, m - 1) * (k - 1));
                        m++;
                }
                m = 2;
                k++;
        }
        int p = 2, q = 2;
        while (p * q <= N)
        {
                while( p * q <= N)
                {
                        if (( p * q == N) && (Prime(p)) && (Prime (q) ))
                                return (( p - 1) * (q - 1 ));
                        q++;
                }
                q = 2;
                p++;
        }
```

```
    long e = 1;
    int n = N;
    for (int i = 2; i < N / 2; i++)
    {
        int j = 0;
        while (n % i == 0 && n > 1)
        {
            j++;
            n /=i;
        }
        if(j!=0)
            e *= (long)Math.Pow(i,j-1)*(i-1);
    }
    return(e);
}

static void Main()
{
    int n = int.Parse(Console.ReadLine());
    Console.WriteLine(Euler(n));
    Console.ReadKey();
}
```