```
>
```

Want to treat "this is a phrase" with characters like "th" "is" " i" "s " ... etc

```
> with(StringTools):
> Alphabet:=cat("\n\t",Select(IsPrintable,convert([seq(i, i=1..255)
  ], bytes))):
  p:=length(Alphabet);
```

$$p := 97 \tag{1}$$

```
> StringToList := proc (text::string)
  local i;
  global Alphabet;
  [seq(SearchText(text[i], Alphabet)-1, i = 1 .. length(text))];
  end proc:

  ListToString := proc (numlist::(list(nonnegint)))
   local i;
   global Alphabet;
   cat(seq(Alphabet[numlist[i]+1], i = 1 .. nops(numlist)))
   end proc:
> StringToVect:=proc( t::string, n::posint)
    local L, s;
    s:=t;
    while ( modp(length(s),n) <> 0) do
        s:=cat(s,"X");
    od;
    L:=StringToList(s);
    [seq(<seq(L[i+j], j=0..n-1)>, i=1..nops(L)-1,n)];
  end:
  VectToString:=proc( vlist )
    ListToString(map(x->op(convert(x,list)),vlist))
  end:
> AffineMatCrypt:= proc(text::string, A::Matrix, b::Vector)
    local v,n;
    n:=Dimension(A)[1];
    v:=StringToVect(text,n);
    VectToString([seq( modp(A.v[i]+b,p), i=1..nops(v))]);
  end:
> AffineMatDecrypt:= proc(text::string, A::Matrix, b::Vector)
    local v,n;
    n:=Dimension(A)[1];
    v:=StringToVect(text,n);
    VectToString([seq( modp(A^(-1).(v[i]-b),p), i=1..nops(v))]);
  end:
```

writing 1065 means 1*1000 + 0*100+6*10 + 5*1.

write 5 in binary.

1*2^0 + 0*2^1 + 1*2^2 === 101 (binary)

```
> convert(5, binary);
```

(2)

```
                                  101                                  (2)
> convert(13, binary);
                                 1101                                  (3)
> convert(5,hexadecimal);
                                   5                                   (4)
> convert(17,hexadecimal);
                                  11                                   (5)
> convert(13,hexadecimal);
                                   D                                   (6)
> convert(1234,hexadecimal);
                                  4D2                                  (7)
> convert(10, base, 4);
                                [2, 2]                                 (8)
```

convention in maple is backwards:

```
> convert(11, base, 4);
                                [3, 2]                                 (9)
> convert(1234, base, 16);
                              [2, 13, 4]                               (10)
> convert([2,13,4], base, 16, 10);
                              [4, 3, 2, 1]                             (11)
```

```
> StringToList("four");
                             [72, 81, 87, 84]                          (12)
> [72+97*81, 87+84*97];
                             [7929, 8235]                              (13)
> 97*97;
                                9409                                   (14)
> StringToDigraph:=proc(s::string)
    local l,i;
    l:=StringToList(s);
    if (nops(l) mod 2 <>0) then
      l:=[op(l),0];
     fi:
    [seq( l[i]+l[i+1]*97,i=1..nops(l), 2)];
  end:
> StringToDigraph("four");
                             [7929, 8235]                              (15)
> StringToDigraph("fiver");
                            [7347, 6975, 84]                           (16)
> StringToList("fiver");
                           [72, 75, 88, 71, 84]                        (17)
> convert(StringToList("fiver"), base, p, p^2);
                            [7347, 6975, 84]                           (18)
> convert(StringToList("fiver"), base, p, p^3);
                            [835339, 8219]                             (19)
> c3:=convert(StringToList("fiver"), base, p, p^3);
```

$$c3 := [835339, 8219] \quad (20)$$

```
> ListToString(convert(c3,base,p^3,p));
```

"fiver" (21)

```
> nextprime(12345678901234567890);
```

12345678901234567891 (22)

```
> isprime(12345678111);
```

false (23)

```
> ifactor(12345678111);
```

(3) (13) (31) (1447) (7057) (24)

```
> isprime
  (123456781111111111111111111111111111111111111111111111111111111111
  11111111114444444444444444441);
```

false (25)

```
> ifactor
  (123456781111111111111111111111111111111111111111111111111111111111
  11111111114444444444444444441);
```

Warning, computation interrupted

```
> pr:=17;
```

$pr := 17$ (26)

```
> 1234^16 mod 17;
```

1 (27)

Fermat's Little Theorem: for any prime p and any a which is not a multiple of p, $a^{p-1} = 1 \mod p$

```
> for i from 1 to 50 do
    print(i,modp(i^(pr-1), pr));
  od;
```

1, 1
2, 1
3, 1
4, 1
5, 1
6, 1
7, 1
8, 1
9, 1
10, 1
11, 1
12, 1
13, 1

14, 1
15, 1
16, 1
17, 0
18, 1
19, 1
20, 1
21, 1
22, 1
23, 1
24, 1
25, 1
26, 1
27, 1
28, 1
29, 1
30, 1
31, 1
32, 1
33, 1
34, 0
35, 1
36, 1
37, 1
38, 1
39, 1
40, 1
41, 1
42, 1
43, 1
44, 1
45, 1
46, 1
47, 1
48, 1
49, 1
50, 1