```
> with(StringTools):
> Alphabet := Select(IsPrintable, convert([seq(i,i=1..127)], bytes)
  ) ;
Alphabet :=                                                        (1)
    " !"#$%'()*+,-./0123456789:<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]
    ^_`abcdefghijklmnopqrstuvwxyz{|}~"
> StringToList := proc (str::string)
    global Alphabet;
    return([seq(SearchText(str[i], Alphabet)-1, i = 1 .. length
  (str))]);
  end:
  ListToString := proc (l::list(nonnegint))
    global Alphabet;
    return(cat(seq(Alphabet[l[i]+1], i = 1 .. nops(l))))
  end:

> Caesar:=proc(plain::string, shift::integer)
    local L, S, len;
    global Alphabet;
    len:=length(Alphabet);
    L:=StringToList(plain); # convert to numbers
  ## S:=[seq( (L[i]+shift) mod len, i=1..nops(L))];
    S:=map(n->(n+shift) mod len, L);  # same as above.
    return(ListToString(S));
    end:
> Vignere:=proc(plain::string, key::string)
    local L, shifts, S, len, keylen;
    global Alphabet;
    len:=length(Alphabet);
    keylen:=length(key);
    shifts:=StringToList(key);  #list of shifts.
    L:=StringToList(plain); # convert to numbers
    S:=[seq( (L[i]+shifts[((i-1) mod keylen)+1]) mod len, i=1..nops
  (L))];
    return(ListToString(S));
    end:
>  \
Error, unable to parse
                                   \
> rand( );
                                   395718860534                    (2)
> rand();
                                   193139816415                    (3)
```

```
> randomize( );
                                1394720446                                   (4)
> rand( );
                               605297621301                                  (5)
> randomize(31415);
                                  31415                                      (6)
> rand( );
                               270519376039                                  (7)
> rand( );
                               975308613645                                  (8)
> randomize(31415); rand( ); rand( );
                                  31415
                               270519376039
                               975308613645                                  (9)
> randomize(31416); rand( ); rand( );
                                  31416
                               303119940975
                               285502367145                                 (10)
> DiceRoll := ( ) → rand( ) mod 6 + 1;
                  DiceRoll := ( ) → rand( ) mod 6 + 1                       (11)
> DiceRoll( );
                                    1                                       (12)
> DiceRoll( );
                                    1                                       (13)
> DiceRoll( );
                                    6                                       (14)
> DiceRoll( );
                                    6                                       (15)
> DiceRoll( );
                                    2                                       (16)
> RollDice := rand(1 ..6 );
RollDice := proc( )                                                         (17)
    proc( ) option builtin = RandNumberInterface; end proc(6, 6, 3) + 1
end proc
> seq(RollDice( ), i = 1 ..10);
                         3, 4, 2, 4, 5, 6, 1, 2, 5, 6                       (18)
> seq(RollDice( ), i = 1 ..10);
                         4, 2, 6, 1, 6, 3, 5, 3, 6, 3                       (19)
> randomize(15);
                                    15                                      (20)
```

```
> seq(RollDice( ), i = 1 ..10);
                              1, 6, 5, 6, 1, 5, 4, 4, 6, 6                    (21)
> seq(RollDice( ), i = 1 ..10);
                              2, 6, 1, 3, 5, 2, 6, 4, 5, 6                    (22)
> randomize(15);
                                          15                                  (23)
> seq(RollDice( ), i = 1 ..10);
                              1, 6, 5, 6, 1, 5, 4, 4, 6, 6                    (24)
> FakeOTP:=proc(plain::string, key::posint)
    local L, S, len;
    global Alphabet;
    len:=length(Alphabet);
   randomize(key);

    L:=StringToList(plain); # convert to numbers
    S:=[seq( (L[i]+rand()) mod len, i=1..nops(L))];
    return(ListToString(S));
    end:
  undoFakeOTP:=proc(plain::string, key::posint)
    local L, S, len;
    global Alphabet;
    len:=length(Alphabet);
   randomize(key);

    L:=StringToList(plain); # convert to numbers
    S:=[seq( (L[i]-rand()) mod len, i=1..nops(L))];
    return(ListToString(S));
    end:
> FakeOTP("Invade Crimea now!", 27);
                              "8j}wz-b8d^:@=XfyA?"                            (25)
> undoFakeOTP(%, 27);
                              "Invade Crimea now!"                            (26)
>
>
> Affine:= proc(plain::string, a::integer, b::integer)
    local L, S, len;
    global Alphabet;
    len:=length(Alphabet);
    L:=StringToList(plain); # convert to numbers
    S:=map(x->(a*x+b) mod len, L);  # same as above.
    return(ListToString(S));
    end:
> Affine("Once upon a midnight dreary", 6, 8);
                         "'%!O J(K-'!(2(zbD!bV\E(D9J29c"                      (27)
> f := x→ (6·x + 8) mod 97;
```

$$f := x \to (6\,x + 8) \;\mathbf{mod}\; 97 \qquad \textbf{(28)}$$

> $y = f(x);$ *# solve for x given y.*

$$y = 6\,x + 8 \qquad \textbf{(29)}$$

> $x = \dfrac{(y-8)}{6}$

$$x = \frac{1}{6}\,y - \frac{4}{3} \qquad \textbf{(30)}$$

> $f(3);$

$$26 \qquad \textbf{(31)}$$

> 26 - 8;

$$18 \qquad \textbf{(32)}$$

> $\dfrac{18}{6}$

$$3 \qquad \textbf{(33)}$$

> $f(5);$

$$38 \qquad \textbf{(34)}$$

> $38 - 8;$

$$30 \qquad \textbf{(35)}$$

> $f(0);$

$$8 \qquad \textbf{(36)}$$

> $\dfrac{8-8}{6};$

$$0 \qquad \textbf{(37)}$$

> $\dfrac{1}{2}\,\mathbf{mod}\; 5;$

$$3 \qquad \textbf{(38)}$$

```
> unAffine := proc(plain :: string, a :: integer, b :: integer)
    local L, S, len;
    global Alphabet;
    len := length(Alphabet);
    L := StringToList(plain); # convert to numbers
    S := map(x→((x − b)/a) mod len, L); # same as above.
    return(ListToString(S));
    end:
```

> *Affine*("Twist and shout!", 7, 17);

```
"!XU!  C1|x21!  N JC8"                                    (39)
```

> *unAffine*(%, 7, 17);

```
"Twist and shout!"                                        (40)
```

>

```
> Affine("Twist and shout!", 5, 17);
                              "wh"TY1Y;h1T|@^Y6"                          (41)
> unAffine(%, 5, 17);
                               ".C |'( ("C  '/#)(!"                        (42)
> nextprime(85);
                                      89                                  (43)
> gcd(6, 95);
                                       1                                  (44)
> gcd(5, 95);
                                       5                                  (45)
> Affine := proc(plain :: string, a :: integer, b :: integer)
    local L, S, len;
    global Alphabet;
    len := length(Alphabet);
    if (gcd(len, a) > 1) then
      error(a, " is not relatively prime to length of Alphabet", len);
    end;
    L := StringToList(plain); # convert to numbers
    S := map(x → (a*x + b) mod len, L); # same as above.
    return(ListToString(S));
    end:

> Affine("Oh no!", 5, 0 );
Error, (in Affine) 5,  is not relatively prime to length of
Alphabet, 95
> Affine("Undo this, please", 7, 6);
                             "|m't&8CJ1z&{_.q1."                          (46)
> Affine(%, 1/7, -6/7);
Error, invalid input: Affine expects its 2nd argument, a, to be
of type integer, but received 1/7
> 1/7 mod 95; -6/7 mod 95;
                                      68
                                      67                                  (47)
> Affine("|m't&8CJ1z&{_.q1.", 68, 67);
                              "Undo this, please"                         (48)
>
```