```
> with(StringTools):
> Alphabet := " ABCDEFGHIJKLMNOPQRSTUVWXYZ";
                    Alphabet := " ABCDEFGHIJKLMNOPQRSTUVWXYZ"                    (1)
> Julius := (text, shift) → cat( seq( Alphabet[ SearchText( text[ i], Alphabet) + shift
    mod 27], i = 1 ..length(text))):
> Julius("ORANGE", 3);
                                "RUDQJH"                                         (2)
> Julius("ORANGE", 0);
                                "ORANGE"                                         (3)
> Julius("ORANGE", 5);
                                "TWFSLJ"                                         (4)
> Julius(%, -5);
                                "ORANGE"                                         (5)
> Julius("ORANGE", 12);
Error, (in Julius) invalid range for string subscript
> Julius("ORANGE", 17);
                                "EHRDXV"                                         (6)
> Julius("ZZZ", 0);
Error, (in Julius) invalid range for string subscript
> Julius := (text, shift) → cat( seq( Alphabet[ 1 + (SearchText( text[ i], Alphabet)
    + shift) mod 27], i = 1 ..length(text))):
> Julius("ZZZ", 0);
                                 " "                                             (7)
> Julius("ABC", 0);
                                "BCD"                                            (8)
> Julius := (text, shift) → cat( seq( Alphabet[ 1 + (SearchText( text[ i], Alphabet)
    + shift) mod 26], i = 1 ..length(text))):
> Julius("ZZZ", 0);
                                "AAA"                                            (9)
> Julius := (text, shift) → cat( seq( Alphabet[ (SearchText( text[ i], Alphabet) + shift)
    mod 27], i = 1 ..length(text))):
> Julius("ZZZ", 0);
Error, (in Julius) invalid range for string subscript
> Julius := (text, shift) → cat( seq( Alphabet[ (SearchText( text[ i], Alphabet) + shift)
    mod 27], i = 0 ..length(text))):
> Julius("ZZZ", 0);
Error, (in Julius) invalid range for string subscript
> Julius("A", 0);
Error, (in Julius) invalid range for string subscript
> SearchText("Z", Alphabet);
                                   27                                            (10)
> SearchText("Z", Alphabet) mod 27;
                                                                                 (11)
```

```
                                    0                                    (11)

> (SearchText("Z", Alphabet) mod 27) + 1
                                    1                                    (12)

> ((SearchText("Z", Alphabet) - 1)mod 27) + 1
                                   27                                    (13)

> ((SearchText(" ", Alphabet) - 1)mod 27) + 1
                                    1                                    (14)

> Julius := (text, shift)→cat( seq( Alphabet[ (((SearchText(text[i], Alphabet)
      + shift) - 1)mod 27) + 1], i = 1..length(text))) :

> Julius("Z", 0);
                                  "Z"                                    (15)

>

> Julius("Time flies like an arrow and fruit flies like a banana.", 5);
"YDDDEDDDDDEDDDDEDDEDDDDDEDDDEDDDDDEDDDDDEDDDDEDEDDDD\                  (16)
 DDD"

> Julius(%, -5);
      "TZZZ ZZZZZ ZZZZ ZZ ZZZZZ ZZZ ZZZZZ ZZZZZ ZZZZ Z ZZZZZZZ"         (17)

> Alphabet
      := "ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz.";
Alphabet := "ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz."   (18)

> Julius("Time flies like an arrow and fruit flies like a banana.", 5);
  "YNRJEKQNJXEQNPJEFSEFWWTAEFSIEKWZNYEKQNJXEQNPJEFEGFSFSFE"            (19)

> Julius(%, -5);
      "TIME FLIES LIKE AN ARROW AND FRUIT FLIES LIKE A BANANA "        (20)

> Julius := (text, shift)→cat( seq( Alphabet[ (((SearchText(text[i], Alphabet)
      + shift) - 1)mod length(Alphabet)) + 1], i = 1..length(text))) :

> Julius("Time flies like an arrow and fruit flies like a banana.", 5);
       "YnrjekqnjxeqnpjefsefwwtAefsiekwznyekqnjxeqnpjefegfsfsfE"       (21)

> Julius(%,-5);
            "Time flies like an arrow and fruit flies like a banana."  (22)

> Julius("Time flies like an arrow, fruit flies like a banana.", 5);
         "YnrjekqnjxeqnpjefsefwwtAEekwznyekqnjxeqnpjefegfsfsfE"        (23)

> Julius(%,-5);
              "Time flies like an arrow. fruit flies like a banana."   (24)

> SearchText(",", Alphabet);
                                    0                                    (25)

> -1 mod length(Alphabet);
                                   53                                    (26)

> Alphabet;
     "ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz."          (27)
```

```
> convert(Alphabet, bytes);
```
[65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 32, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 46] **(28)**

```
> convert([32, 46, 97], bytes);
```
" .a" **(29)**

```
> convert([29], bytes);
```
" " **(30)**

```
> convert([seq(i, i = 1 ..127)], bytes);
```
" **(31)**

!"#$%&'()*C ,-./0123456789:;! =O?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}
~• "

```
> IsPrintable("A");
```
true **(32)**

```
> IsPrintable(convert([3], bytes));
```
false **(33)**

```
> convert([3], bytes);
```
" " **(34)**

```
> Select(IsPrintable, convert([seq(i, i = 1 ..127)], bytes));
```
" !"#$%&'()*C ,-./0123456789:;! =O?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\] **(35)**
^_`abcdefghijklmnopqrstuvwxyz{|}~"

```
> Select(IsUpper, convert([seq(i, i = 1 ..127)], bytes));
```
"ABCDEFGHIJKLMNOPQRSTUVWXYZ" **(36)**

```
> Alphabet := Select(IsPrintable, convert([seq(i, i = 1 ..127)], bytes));
```
Alphabet := **(37)**
" !"#$%'()*+,-./0123456789:<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]
^_`abcdefghijklmnopqrstuvwxyz{|}~"

```
> Julius("This is the time and this is the record of the time.", 6);
```
"Znoy&oy&znk&zosk&gtj&znoy&oy&znk&xkiuxj&ul&znk&zosk4" **(38)**

```
> Julius(%, -6 );
```
"This is the time and this is the record of the time." **(39)**

```
>
>
>
> StringToList := proc(str :: string)
    return(str);
  end:
> StringToList("mamma");
```
(40)

"mamma" (40)

```
> StringToList(23);
```
Error, invalid input: StringToList expects its 1st argument, str, to be of type string, but received 23

```
> StringToList("23");
```
"23" (41)

```
> timestwo := proc(x :: integer)
    2·x;
  end:
> timestwo(7);
```
14 (42)

```
> timestwo("rabbit");
```
Error, invalid input: timestwo expects its 1st argument, x, to be of type integer, but received rabbit

```
> timestwo(1.7);
```
Error, invalid input: timestwo expects its 1st argument, x, to be of type integer, but received 1.7

```
> timestwo := proc(x :: numeric)
    2·x;
  end:
> timestwo(1.7);
```
3.4 (43)

```
> timestwo(x);
```
Error, invalid input: timestwo expects its 1st argument, x, to be of type numeric, but received x

```
> StringToList := proc(str :: string)
    global Alphabet;
    l := seq(SearchText(str[i], Alphabet) − 1, i = 1 ..length(str) );
    return(l);
  end:
```
Warning, `l` is implicitly declared local to procedure `StringToList`

```
> StringToList("hello");
```
72, 69, 76, 76, 79 (44)

```
> StringToList := proc(str :: string)
    local l;
    global Alphabet;
    l := [seq(SearchText(str[i], Alphabet) − 1, i = 1 ..length(str) ) ];
    return(l);
  end:
>
> StringToList("hello");
```
[72, 69, 76, 76, 79] (45)

```
> x := 17;
```
x := 17 (46)

```
> trylocal := proc( )
    local x;
    x := 22;
    return(x·4);
  end;
```

$$trylocal := \mathbf{proc}()\ \mathbf{local}\ x;\ x := 22;\ \mathbf{return}\ 4*x\ \mathbf{end\ proc} \quad (47)$$

```
> trylocal( );
```

$$88 \quad (48)$$

```
> x;
```

$$17 \quad (49)$$

```
> tryglobal := proc( )
    global x;
    x := 22;
    return(x·4);
  end;
```

$$tryglobal := \mathbf{proc}()\ \mathbf{global}\ x;\ x := 22;\ \mathbf{return}\ 4*x\ \mathbf{end\ proc} \quad (50)$$

```
> x;
```

$$17 \quad (51)$$

```
> tryglobal( );
```

$$88 \quad (52)$$

```
> x;
```

$$22 \quad (53)$$

```
> l := StringToList("hi,there");
```

$$l := [72, 73, 12, 84, 72, 69, 82, 69] \quad (54)$$

```
> convert(l, bytes); # This is wrong.
```

$$\text{"HI\ \ THERE"} \quad (55)$$

```
> Alphabet[72];
```

$$\text{"g"} \quad (56)$$

```
> Alphabet[72 + 1];
```

$$\text{"h"} \quad (57)$$

```
> ListToString := proc(l :: list)
    global Alphabet;
    cat(seq(Alphabet[l[i] + 1], i = 1 ..nops(l)));
  end:
> ListToString(l);
```

$$\text{"hi,there"} \quad (58)$$

```
> StringToList("Anything you can write. Yeah!!!!");
```

$$[33, 78, 89, 84, 72, 73, 78, 71, 0, 89, 79, 85, 0, 67, 65, 78, 0, 87, 82, 73, 84, 69, 14, 0, 57, 69, 65, 72, 1, 1, 1, 1] \quad (59)$$

```
> ListToString(%);
```

$$\text{"Anything you can write. Yeah!!!!"} \quad (60)$$

```
>
```