Numbers, Sets, and Lists

Numbers

There are several essentially different ways to represent a number in the computer.

Exact (arbitrarily long) representation of integers (and rationals)

This is how Maple treats integers and rationals.

```
> for k from 1 to 10 do factorial(k); end do;
```

1
2
6
24
120
720
5040
40320
362880
3628800 (1)

- arithmetic operations +,-,*,/ are exact
- size only limited by computer memory
- implemented as a dynamic data structure (lists of memory **words**)
- not dircetly available in most programming languages

```
> factorial(10000)/factorial(9999)
```

10000 (2)

```
> solve(x*factorial(9999) = 10000, x)
```

1/ (3)

28462596809170545189064132121198688901480514017027992307941799947\
27441134000376444377299078675778477 5[...35452 digits...]
00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000

Representation of numbers by symbols

This is not available in standard programming languages.

```
> sqrt(2)
```

$\sqrt{2}$ (4)

```
> sqrt(2)*sqrt(2)
```

2 (5)

```
> sin(Pi/4)
```

$\frac{1}{2}\sqrt{2}$ (6)

```
> solve(x^5 - 4*x + 2, x)
```

RootOf($\_Z^5 - 4\_Z + 2$, *index* = 1), RootOf($\_Z^5 - 4\_Z + 2$, *index* = 2), (7)

$$\text{RootOf}\left(\_Z^5 - 4\,\_Z + 2,\ index = 3\right),\ \text{RootOf}\left(\_Z^5 - 4\,\_Z + 2,\ index = 4\right),$$
$$\text{RootOf}\left(\_Z^5 - 4\,\_Z + 2,\ index = 5\right)$$

Representation of natural numbers with fixed word size

- size limited by computer's **word size**, typically a power of 2
- arithmetic operations +,-,*,/ are exact unless overflow occurs
- standard integer arithmetic for most programming languages

> $WordSize := 32$:

> $Modulus := 2^{WordSize}$

$$Modulus := 4294967296 \quad (8)$$

> $(2+2)$ **mod** $Modulus$

$$4 \quad (9)$$

> factorial(10) **mod** $Modulus$

$$3628800 \quad (10)$$

> $b := \text{factorial}(13)$

$$x := 6227020800 \quad (11)$$

> **if** $(b < Modulus)$ **then** $print(\text{"small enough"})$; **else** $print(\text{"too big"})$; **end if**;

$$\text{"too big"} \quad (12)$$

> b **mod** $Modulus$

$$1932053504 \quad (13)$$

> $1-2$ **mod** $Modulus$

$$4294967295 \quad (14)$$

Representation of negative numbers left for exercise.

Approximate representation of reals by floating point numbers

- size can or cannot by limited by word size
- in Maple the size is not limited
- arithmetic operations **not** exact

> $fsolve(a \cdot \text{factorial}(9999) = 10000,\ a)$

$$3.513382868\ 10^{-35652} \quad (15)$$

> $2.13579 \cdot 10^3$

$$2135.79000 \quad (16)$$

Later: Example computations to show the limits of floating point computations (even with arbitrarily long representation).

Complex numbers

- internally represented as pairs of integer/rational/floating point numbers
- symbolically represented numbers may be tagged as complex

> $\text{sqrt}(-1)$

$$I \quad (17)$$

> $(3-4 \cdot I) \cdot \left(2 + \frac{1}{3} \cdot I\right)$

$$\frac{22}{3} - 7\,I \quad (18)$$

> $solve(x^2 = -1,\ x)$

$$I,\ -I \quad (19)$$

```
> I := -34
Error, illegal use of an object as a name
                                  I := -34
> fsolve(x^2 = -1, x)
> fsolve(x^2 = -1, x, complex)
                          -1.000000000 I, 1. I                    (20)
```

Sets, Lists, and Sequences

```
> {2, 3, 4, 5}
                          {2, 3, 4, 5}                            (21)
> {2, 5, 4, 3, 3, 5}
                          {2, 3, 4, 5}                            (22)
> {3 + I, factorial(6), I, 4, 5}
                       {4, 5, 720, I, 3 + I}                      (23)
> {2, 3} union {4, 5}
                          {2, 3, 4, 5}                            (24)
> seq(1..10)
                       1, 2, 3, 4, 5, 6, 7, 8, 9, 10              (25)
> {seq(1..10)} intersect {2, 3, 5, 7, 11, 13}
                          {2, 3, 5, 7}                            (26)
> with(numtheory):
> divisors(120)
                 {1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 24, 30, 40, 60, 120}   (27)
```

A natural number is **perfect** if it is the sum of its divisors (except for itself).

```
> isPerfect := proc(n)
    local DivisorsOfN, k;
    DivisorsOfN := divisors(n);
    if (sum(DivisorsOfN[k], k = 1..nops(DivisorsOfN)) = 2·n) then
      return true;
    else
      return false;
    end if;
  end proc:
> isPerfect(6)
                             true                                 (28)
> isPerfect(7)
                             false                                (29)
> ListPerfectNumbers := proc(n)
    local M, k;
    M := [ ];
    for k from 1 to n do
      if (isPerfect(k)) then M := [op(M), k]; end if;
    end do;
    return M;
  end proc:
> ListPerfectNumbers(1000)
```

[6, 28, 496] **(30)**

```
> FilterForProperty := proc(L, Predicate)
   local M, k;
   M := [ ];
   for k from 1 to nops(L) do
    if Predicate(L[k]) then M := [op(M), L[k]]; end if;
   end do;
   M;
  end proc:
> FilterForProperty( [seq(1 ..100) ], isPerfect)
```

[6, 28] **(31)**