1. Niech B bedzie liczba naturalna wieksza od 1. Wykazac, ze kazda niezerowa liczba rzeczywista x ma jednoznaczne przedstawienie w postaci znormalizowanej $x = smB^C$, gdzie s jest znakiem liczby x, c – liczba calkowita (cecha), a m – liczba z przedzialu $[1, B)$, zwana mantysa.

1. istnienie:

Niech $0 \neq x \in \mathbb{R}$. Wtedy istnieje c takie, ze $B^c \leq |x| < B^{c+1}$, czyli $c = \lfloor \log_B |x| \rfloor$. Niech $m = \frac{|x|}{B^c}$, wtedy $|x| = B^2 \cdot m$. W koncu, niech $s = \frac{|x|}{x}$. Mamy $x = sB^c m$.

2. jedynosc:

   A. jedynosc s jest oczywista

   B. jedynosc c: zalozmy, nie wprost, ze istnieja c_1, c_2 takie, ze

$$x = sB^{c_1} m = sB^{c_2} m'$$

Wtedy

$$B^{c_1} m = B^{c_2} m'$$

Jesli $m = m'$ oczywiste. W przeciwnym wypadku

$$c_1 \log_B m = c_2 \log_B m'$$

mozemy zalozyc, ze $c_1 < c_2$ oraz $(\exists\, k \in \mathbb{N})\; c_1 + k = c_2$, czyli

$$c_1 \log_B m = (c_1 + k) \log_B m'$$

$$0 = k \log_B m'$$

W takim razie albo $m' = 1$, wtedy $x = 2^c_1$, albo $k = 0$, czyli $c_1 = c_2$.

   C. jedynosc m: zalozmy, nie wprost, ze istnieja m_1, m_2 takie, ze (...), wtedy

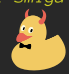
$$x = sB^c m_1 = sB^c m_2.$$

c jest jedyne, gdyz $c = \lfloor \log_B |x| \rfloor$ i to dzialanie ma jednoznaczny wynik. Czyli

$$sB^c m_1 = sB^c m_2$$

$$m_1 = m_2$$

i śmiga



2. Ile jest liczb zmiennopozycyjnych w arytmetyce double w standardzie IEE754?

Przy 64 bitach mamy 2^{64} mozliwosci ich zapalenia. Liczby NaN to liczby majace wszystkie bity w mantysie zapalone, a takich jest 2^{53}. To daje nam $2^{64} - 2^{53}$ liczb, ale 0 jest reprezentowane na dwa sposoby, wiec wystarczy $2^{64} - 2^{53} - 1$.
Liczby subnormalne maja na pierwszym miejscu mantysy 0, podczas gdy cala reszta zaczyna mantyse od 1 i one juz sie wliczaja.

3. Czesc rozw w pliku .jl

```
function frst_exp(x, s, t, r)
    ret = zero(x)
    ret = (x^3) - (s*(x^2)) + t*x - r
    print("␣␣", typeof(x), "␣wynik:␣", ret, "\n")
end

```

```julia
function snd_exp(x, s, t, r)
    ret = zero(x)
    ret = ((x - s) * x + t) * x - r
    print("␣␣", typeof(x), "␣wynik:␣", ret, "\n")
end

function rel_error(val, exp, mess)
    bez = zero(val)
    if val > exp
        bez = val - exp
    else
        bez = exp - val
    end

    println("Blad␣wzgledny␣", mess, "␣wynosi:␣", bez / exp)
end
```

-14.636489 - wartosc prawidlowa

Float16	-0.003987568330082385	-0.0002511872895199931
Float32	-7.760455081662309e-7	-5.931504907397523e-8
Float64	-4.85459822885188e-16	0

Zad 4. Za dluga tresc

$$\sum_{k=t+2}^{\infty} \frac{1}{2^k} = 2\frac{1}{2^{t+2}} = 2^{-t-1}$$

$$|rd(x) - x| = |s(1 + \sum_{k=1}^{\infty} e_{-k}2^{-k})2^c - s(1 + \sum_{k=1}^{t} e_{-k}2^{-k} + e_{-t-1}2^{-t})2^c| =$$

$$= 2^c|\sum_{k=1}^{\infty} e_{-k}2^{-k} - \sum_{k=1}^{t} e_{-k}2^{-k} - e_{-t-1}2^{-t}| =$$

$$= 2^c|\sum_{k=t+1}^{\infty} e_{-k}2^{-k} - e_{-t-1}2^{-t}| \leq$$

$$\leq 2^c|\sum_{k=t+1}^{\infty} 2^{-k} - e_{-t-1}2^{-t}| =$$

$$= 2^c|\sum_{k=t+2}^{\infty} 2^{-k} + e_{-t-1}2^{-t-1} - e_{-t-1}2^{-t}| =$$

$$= 2^c|\sum_{k=t+2}^{\infty} 2^{-k} + e_{-t-1}(\frac{1}{2^{t+1}} - \frac{1}{2^t})| =$$

$$= 2^c|\frac{1}{2^{t+1}} - \frac{e_{-t-1}}{2^{t+1}}| \leq$$

$$\leq 2^c \cdot 2^{-t-1} = 2^c \cdot u$$

$$\frac{|rd(x) - x|}{|x|} = \frac{2^c|m - \overline{m}|}{2^c|m|} = \frac{|\sum_{k=t+1}^{\infty} e_{-k}2^{-k} - \ldots|}{|1 + \sum_{k=1}^{\infty} e_{-k}2^{-k}|} \leq \frac{u}{1} = u$$

5. nah

$$\frac{|\overline{m} - m|}{|m|} \leq \frac{u}{1 + u}$$

$$(1 + u)|\overline{m} - m| \leq u|m|$$

Z poprzedniego zadania wiemy, ze $|\overline{m} - m| \leq u$. Wystarczy wiec, ze rozwazymy dwa przypadki:
1. $(1 + u) \leq m$ – oczywiste
2. $(1 + u) > m$

$$\begin{aligned} u &> m - 1 \\ \overline{m}u &> m\overline{m} - \overline{m} > m - \overline{m} \\ \overline{m}u + mu &> m - \overline{m} + mu \\ mu &> m - \overline{m} + mu - \overline{m}u \\ mu &> m(1 + u) - \overline{m}(1 + u) \\ mu &> (1 + u)(m - \overline{m}) \\ \frac{u}{1 + u} &> \frac{m - \overline{m}}{m} \end{aligned}$$

6. w pliiikuuuuuu

```
using Base

function to_number(str::AbstractString)
    if length(str) <= 64
        bias = Float64(2^10 - 1)

        ret = Float64(0)
        cech = Float64(0) # 11
        man = Float64(1) # 53

        c = Float64(1)

        for i = 1:11
            cech += c * (Float64(str[13 - i]) - Float64(48))
            c *= Float64(2)
        end

        c = Float64(0.5)

        for i = 1:52
            man += c * (Float64(str[i+12]) - Float64(48))

            c /= Float64(2)
        end

        cech -= bias

        ret = man * (2 ^ cech)

        if str[1] == '1'
            ret *= Float64(-1)
        end

        return ret
    else
        print("this is not a representation of a Float64")
        return NaN
    end
end
```

7.

Liczba 1.000000057228997

```
function fl()
```

```
    x = one(Float64)

    while x < 2.0
        if x * (1/ x) != 1
            println(x)
            println(x * (1 / x))
            break
        end
        x += eps(x)
    end
end
```