Homework no. 1

1. Find the smallest positive number $u > 0$, written as a negative power of 10, $u = 10^{-m}$, which satisfies the property:

$$1.0 +_c u \neq 1.0.$$

In the above relation $+_c$ denotes the computer implemented addition operation. The number u is known as *machine precision.*

2. 2. Operation $+_c$ is *non-associative*: consider the numbers $a = 1.0$, $b = u/10$, $c = u/10$, where u is the above computed machine precision. Verify that the computer addition operation is non-associative, i.e.:

$$(a +_c b) +_c c \neq a +_c (b +_c c).$$

Find an example that shows the computer multiplication operation is also non associative.

3. **Approximation of the Tangent Function**

Implement the following two methods that compute approximations of the values for the tangent trigonometric function: the continued fractions method and a polynomial approximation. Apply these methods for approximating the tan function for arguments $x \in (-\frac{\pi}{2}, \frac{\pi}{2})$. For x values that are not in the interval $(-\frac{\pi}{2}, \frac{\pi}{2})$ one uses the periodicity of the tangent function and the anti-symmetry $\tan(x) = -\tan(-x)$. The values of x that are equal to $k\frac{\pi}{2}(k \in \mathbb{Z})$ must be analyzed separately.

Compare the value of the tangent function obtained withe the two below described methods with the value provided by the tangent function implemented in the mathematical library of the programming language you are using. Display $|\tan(x) - \text{my\_tan}(x)|$.

Generate 10.000 numbers in $(-\frac{\pi}{2}, \frac{\pi}{2})$, $\{x_i \in (-\frac{\pi}{2}, \frac{\pi}{2}); i = 1, 10.000\}$. Employ the two methods (described below) for computing the value of the tangent function . Compare these methods taking into account the computation error ($|\tan(x) - \text{my\_tan}(x)|$) and the computing time (for all the 10.000 values of the argument).

Tangent Function Approximation Using Continued Fractions

A continued fraction has the following form:

$$f = b_0 + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cfrac{a_3}{b_3 + \cfrac{a_4}{b_4 + \cfrac{a_5}{b_5 + \cdots}}}}}$$

or, for saving writing space, one prefers the notations:

$$f = b_0 + \frac{a_1}{b_1 +} \; \frac{a_2}{b_2 +} \; \frac{a_3}{b_3 +} \; \frac{a_4}{b_4 +} \; \frac{a_5}{b_5 +} \; \cdots$$

In the above formulae the sequences a_n and b_n can be function of x.

The tangent function tan can be represented using a continued fraction as follows:

$$\tan\, x = \frac{x}{1 +} \; \frac{(-x^2)}{3 +} \; \frac{(-x^2)}{5 +} \; \frac{(-x^2)}{7 +} \cdots$$

How can be numerically approximated a continued fraction?

$$f \approx f_n = \frac{A_n}{B_n}$$

where A_n and B_n are computed using the recurrence relations:

$$\begin{aligned} &A_{-1} = 1 \quad , \quad A_0 = b_0 \\ &A_j = b_j A_{j-1} + a_j A_{j-2} \quad , \quad j = 1, 2, \ldots, n \end{aligned}$$

$$\begin{aligned} &B_{-1} = 0 \quad , \quad B_0 = 1 \\ &B_j = b_j B_{j-1} + a_j B_{j-2} \quad , \quad j = 1, 2, \ldots, n \end{aligned}$$

The best method to evaluate continued fractions seems to be *modified Lentz's method*. Let:

$$C_j = \frac{A_j}{A_{j-1}} \quad , \quad D_j = \frac{B_{j-1}}{B_j}$$

we compute f_j using the formula:

$$f_j = C_j D_j f_{j-1}.$$

For C_j and D_j one has the following recurrence relations:

$$C_j = b_j + \frac{a_j}{C_{j-1}} \quad ,$$

$$D_j = \frac{1}{b_j + a_j D_{j-1}} \ .$$

The initial values for the above relations are:

$$C_0 = b_0 \quad , \quad D_0 = 0 \quad , \quad f_0 = b_0.$$

To avoid division by 0 in the above computations, the zero denominators will be replaced with a very small value ($small = 10^{-12}$). One computes f_j as long as the absolute value of the differences between two consecutive elements of the sequence f is greater than a given $\epsilon = 10^{-p}$. The number ϵ represents the computation precision for approximating the tangent function tan. This number is introduced from keyboard and is a input parameter of the tan function :

double my_tan(double x, double ϵ).

The modified Lentz's algorithm has the following structure:

$f_0 = b_0$; $small = 10^{-12}$;
if $(f_0 = 0)$ **then** $f_0 = small$;
$C_0 = f_0$;
$D_0 = 0$;
$j = 1$;
do
  $D_j = b_j + a_j D_{j-1}$;
  if $(D_j = 0)$ **then** $D_j = small$;
  $C_j = b_j + \frac{a_j}{C_{j-1}}$;
  if $(C_j = 0)$ **then** $C_j = small$;
  $D_j = \frac{1}{D_j}$;
  $\Delta_j = C_j \, D_j$;
  $f_j = \Delta_j \, f_{j-1}$;
  $j = j + 1$;
while $(|\Delta_j - 1| \geq \epsilon)$;

In the above algorithm, for computing the sequences C_j, D_j Δ_j and f_j it is not necessary to declare vectors/arrays in your program. You can use only one variable for each sequence (C, D, Δ, f), this variable is updated in each step of the algorithm.

Tangent Function Approximation Using Polynomials

For approximating the tangent function one can use the following polynomial that was deduced using the MacLaurin series:

$$\tan(x) \approx x + \frac{1}{3}x^3 + \frac{2}{15}x^5 + \frac{17}{315}x^7 + \frac{62}{2835}x^9 = x + P(x^2)x^3$$

To reduce computing time, the coefficients that appear in the above formula can be pre-computed and then used in the function that calculates the value of the above polynomial. Do not write a separate function for computing the polynomial P, use the above formula directly in the function that computes the approximative value of the tangent function.

```
my_tan (x)
c1 = 0.33333333333333333;
c2=0.133333333333333333;
c3=0.053968253968254;
c4=0.0218694885361552;
x_2=x*x;
x_3=x_2*x;
x_4=x_2*x_2;// x_4=x_3*x;
return ...
```

You will find that this approximation formula works better if $x \in (-\frac{\pi}{4}, \frac{\pi}{4})$. To solve this problem of reducing the input parameter $x \in (-\frac{\pi}{2}, \frac{\pi}{2})$ to the interval $[-\frac{\pi}{4}, \frac{\pi}{4}]$, one can use the anti-symmetry property and the following relation:

$$\tan(x) = \frac{1}{\tan(\frac{\pi}{2} - x)} \quad , \quad x \in [\frac{\pi}{4}, \frac{\pi}{2}).$$