Homework Assignment 2

1) Submit as `Assignment_02_1.cpp` in the template according to the instructions.

Newton's method is perhaps the simplest iterative method for finding a zero of a function. The method works as follows. Given an initial approximate value x_0 for a solution to the equation $f(x) = 0$, the next approximate value x_1 is computed by first expanding $f(x)$ in Taylor series up to first order

$$f(x) \approx f(x_0) + f'(x_0)\,(x - x_0)$$

at x_0, where $f'(x_0) = \left.\frac{df}{dx}\right]_{x_0}$, and then solving the linear equation

$$f(x_0) + f'(x_0)\,(x_1 - x_0) = 0$$

The result is

$$x_1 = x_0 - f(x_0)/f'(x_0)$$

Naturally, this result will not be very accurate, but the process can be repeated. Given the n-th approximate solution x_n, the next approximate value x_{n+1} is computed using the same formula

$$x_{n+1} = x_n - f(x_n)/f'(x_n)$$

It is known that if the initial approximate value x_0 is *sufficiently close* to a zero of the function $f(x)$, then the sequence $x_0, x_1, x_2, \ldots$ converges to the zero quadratically (which is very fast).

A. Write a program to find a solution of the equation

$$\cos(x) = 0$$

using 32 bits real types (i.e. **`float`**).

B. Devise a termination condition that insures that the solution will be accurate to the precision of the computation.

C. Observe what happens if you ask for a more precise solution, by insisting that $f(x_n) = 0$ exactly, as the termination condition.

2) Submit as `Assignment_02_2.cpp` in the template according to the instructions.

Consider the following iterative scheme, which can be used to compute π as

$$\pi = \lim_{n\to\infty} a_n = \lim_{n\to\infty} b_n$$

(http://www.cs.umb.edu/~offner/files/pi.pdf),

where $a_0 = 2\sqrt{3}$, $b_0 = 3$,

$$a_{k+1} = \frac{2a_k b_k}{a_k + b_k}$$

and

$$b_{k+1} = \sqrt{a_{k+1} b_k}$$

A. Implement this method using a **`while`** loop and then using a **for** loop.

B. Given a specified maximum number of digits, determine a stopping condition that computes π to the specified number of digits.

C. If the specified number of digits is larger than the maximum number or negative, determine a stopping condition that computes π to the maximum number of digits that are representable in a real number of the given type (**float** and **double**).

3) Submit as `Assignment_02_3.cpp` in the template according to the instructions.

A. Show that the following code segment:

```
// the input value
double x=0.5;
// number of terms/iterations
int n=100;
int k = 0;
double fact_k = 1.0;
double x_k = 1.0;
double exp_x_1 = 1.0; // output value approximates exp(x)
while(++k<=n) {
  x_k *= x;
  fact_k *= k;
  exp_x_1 += x_k/fact_k;

  // add print statement here
}
// print final value, real value, and relative error as a percentage
```

approximates the value of e^x based on the power series expansion

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} = \lim_{n\to\infty} \sum_{k=0}^{n} \frac{x^k}{k!}$$

B. Implement a program that reads the values of **x** and **n** from the command line and print out the result of the computation. You can use the library functions

```
double atof (const char* str);
int atoi (const char * str);
```

defined in the system header file **`stdlib.h`** to parse the command line strings. Print a "usage" message if the number of command line arguments is not sufficient and quit.

Add a print statement within the loop to report the current approximation and the error between the current approximation and the correct value for each value of the variable **`i`**. Use the library function

```
double exp(double x);
```

defined in the system header file **`math.h`** to compute the correct value.

C. In the same loop, add statements to also approximate the value of e^x based the following identity

$$e^x = \lim_{n\to\infty}\left(1 + \frac{x}{n}\right)^n$$

Modify the print statements to report the two approximations on the same line.

D. Which of the two approximations converge faster? Which one of the two methods has lower complexity?