Programming Project 02

This assignment is worth 20 points and must be **completed and turned in before 11:59 on Monday, February 1st , 2016.**

Assignment Overview
This assignment will exercise your ability to utilize control statements for C++ and to write some simple functions.

Background
Prime numbers, those numbers who have as integer divisors only the value 1 and the number itself, are a type of number that has been known since antiquity (Euclid's Elements for example, circa 300 BC). Large prime numbers can be useful in areas such as cryptography, but most study of primes is more "pure" than practical.

One question asked is "how many" primes are in some range. This is a difficult question. Gauss was one of the first to estimate the number of primes less than some number `n` using the logarithmic integral function, often referred to as `Li(n)`. Though it does not yield an exact number, it yields a very good approximation of the count, especially as the counts get larger. The table below (from https://plus.maths.org/content/prime-number-lottery) shows how close his estimate gets as `n` gets large.

N	**Number of primes $\Pi(N)$ from 1 up to N.**	**How far Gauss's guess Li(N) overestimates the number of primes less than N: Li(N)- $\Pi(N)$**	**Percentage error: $\frac{\mathrm{Li}(N)-\Pi(N)}{\Pi(N)} \times 100$**
100	25	5	20
1,000	168	10	5.95
10,000	1,229	17	1.38
100,000	9,592	38	0.396
1,000,000	78,498	130	0.166
10^7	664,579	339	0.051
10^8	5,761,455	754	0.0131
10^9	50,847,534	1,701	0.00335
10^{10}	455,052,511	3,104	0.000682
10^{11}	4,118,054,813	11,588	0.000281
10^{12}	37,607,912,018	38,263	0.000102
10^{13}	346,065,536,839	108,971	0.0000299
10^{14}	3,204,941,750,802	314,890	0.00000983
10^{15}	29,844,570,422,669	1,052,619	0.00000353
10^{16}	279,238,341,033,925	3,214,632	0.00000115

More on the logarithmic integral at https://en.wikipedia.org/wiki/Logarithmic_integral_function
`Li(x)` is actually the equation:

$$Li(x) = \int_{t=0}^{x} \frac{dt}{ln(t)}$$

where `ln(t)` is the natural logarithm.

However, we don't have any "equipment" to do definite integrals in C++, but we can estimate the value of the integral using a series. One series that converges fairly quickly is due to Ramnujan, one of the greatest mathematical minds of 20th century. Here is his series:

$$Li(x) = \gamma + ln(ln(x)) + \sqrt{(x)} \sum_{n=1}^{\infty} \left(\frac{(-1)^{n-1} * (ln(x))^n}{n! * 2^{n-1}} * \left(\sum_{k=0}^{\lfloor \frac{n-1}{2} \rfloor} \frac{1}{2k+1} \right) \right)$$

factorial of n

`term` function

Things to note:

- The symbol γ represents a special constant called the Euler-Mascheroni gamma constant and it has the `double` value: 0.577215664901532 . Lovely isn't it? See https://en.wikipedia.org/wiki/Euler%E2%80%93Mascheroni_constant for details.
- `n!` is the factorial function (see https://en.wikipedia.org/wiki/Factorial).
- The special brackets $\lfloor exp \rfloor$ represent the floor function, the **rounding down** of whatever exp is to a float with a fractional part of 0.
- We have marked off one of the inset summations as a function called `term`.
- Clearly we cannot do a sum to infinity, so we will have to limit our sum to a more reasonable number, giving us a better approximation as that summation number gets bigger.

Project Description / Specification

For this program we will do re-directed input so ***no prompt strings***. See the example below. The required input values are:

- The number of test cases that the program will solve
- For each test case, two numbers: the value x and the iteration number limit (the number we will iterate to instead of ∞).

The sample file `inputs.txt` provides an example of the required input.

For output, your program will provide:

- For each test case on a separate line, the two input values, x and the interation number limit, and the resulting `Li(x)` value using the formula above
- Formatting matters here, make it look nice (mine does).

Requirements

1. No checks on validity of inputs required
2. Write a function `factorial` that takes a single arugment `long n` and returns a `double`, the factorial of n. Do this **iteratively, not recursively**. Easily found on the web
3. Write a function `term` that takes a single `long n` and returns a `double`, the result of the summation indicated in the `term` function section of the series marked above.
4. Write the `main` function to perform all of the tasks indicated, including using the factorial and term function to make the calculation.
5. Place all code in a **single file** named **proj02.cpp**

Deliverables

proj02.cpp -- your source code solution (remember to include your section, the date, project number and comments).

1. Please be sure to use the specified file name, i.e. "**proj02.cpp**"
2. Save a copy of your file in your CS account disk space (H drive on CS computers).
3. Electronically submit a copy of the file.

Notes

1. In the `cmath` include file you will find the following functions useful. Look them up for more details:
    a. `log` : the natural logarithm (the `ln` in the series equation)
    b. `sqrt` : the square root function
    c. `pow` : raise a value to a power
    d. `floor` : the floor function
2. In `iomanip` look at `setw` to get column output. `setw` sets the width of the output ***that directly follows*** will occupy. `setw` is different in that it does not set state for all output. `setw` only sets the width for the next output. Look it up.
3. The `factorial` function works with `doubles` and not `longs` because 20! is the biggest number (even at 64 bits) we can represent. Thus any iteration number limit over 20 would overflow a `long`. With a `double` we can at least do more than 20 iterations.
    a. This also means that your factorial function is not fully accurate. Ah well.
4. We picked this series because it converges quickly. That is, after not-too-many iterations the answer is essentially unchanged. That is good given all the inaccuracies that might get introduced (by the series, by the factorial, etc.).

Example Output