CPS 5401 Fall 2018
Lab Assignment 1
Due 11:59pm Monday, October 8

`C` Lab Assignment: Bessel function of the first kind

Submission Requirement:

You must complete this assignment independently. To submit, create a folder called "`Lab1`" in your Bitbucket repository that is shared with the TA and the instructor. Put all your source codes as well as a `Makefile` and optionally a `readme.txt` in this folder. Make sure all the changes are committed on the local machine and pushed to the remote Bitbucket repository.

The TA will pull your work from Bitbucket, and only look into the folder "`Lab1`" for this assignment. Particularly, typing

```
make
```

in the folder "`Lab1`" should be sufficient to build the code and create the executable. You may put consice instructions of how to use your code in `readme.txt`.

Bitbucket is the only way the submission is accepted, no exception. Make sure that you have your Bitbucket account setup properly ahead of time. No extension will be made in any circumstances, even if the reason is a last-minute power outage or network disruption. So please start early on the lab assignment; and remember that even when you're still working on the codes, you can always commit and push the partial work to the Bitbucket. The total score for this assignment is 25 points.

Problem:

We'll write a program to compute the Bessel functions of the first kind with non-negative integer indices: J_n , $n = 0, 1, \cdots$:

$$J_n(x) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m!(m+n)!} \left(\frac{x}{2}\right)^{2m+n} .$$

The program needs to take three inputs from the command line: the index n, the argument x, and a tolerance $\varepsilon > 0$. Upon execution (with the proper inputs), the program should return an approximation α to $J_n(x)$. In particular, α is the sum of the first M terms of the Taylor series expansion of J_n:

$$\alpha = \sum_{m=0}^{M-1} \frac{(-1)^m}{m!(m+n)!} \left(\frac{x}{2}\right)^{2m+n} ,$$

where M is determined by:

$$\left| \frac{(-1)^M}{M!(M+n)!} \left(\frac{x}{2}\right)^{2M+n} \right| < \varepsilon .$$

Try to design your program from the user's point of view. For example on the one hand, if the users use reasonable input values for n, x, and ε, they expect the program to return reasonably accurate results. On the other hand, the user may accidentally choose a very large value for ε, the program should not break down due to such extreme situations; it may even kindly remind the user that the tolerance is too large. Use your imagination to cook up other scenarios and try to write a robust program.