CSCI 1312 (Introduction to Programming for Engineering), Fall 2018

Homework 5

Credit: 40 points.

1 Reading

Be sure you have read (or at least skimmed) the assigned readings from chapter 6.

2 Programming Problems

Do the following programming problems. You will end up with at least one code file per problem. Submit your program source (and any other needed files) by sending mail to bmassing@cs.trinity.edu with each file as an attachment. Please use a subject line that mentions the course and the assignment (e.g., "csci 1312 hw 5" or "CS1 hw 5"). You can develop your programs on any system that provides the needed functionality, but I will test them on one of the department's Linux machines, so you should probably make sure they work in that environment before turning them in.

1. (20 points) *NOTE* that part of the point of this problem is for you to practice using `while` loops, so you must use at least one to get full credit, and I strongly recommend that you do all the needed repetition using `while`.

   Newton's method for computing the square root of a non-negative number x starts with an initial guess r_0 and then repeatedly refines it using the formula

   $$r_n = (r_{n-1} + (x/r_{n-1}))/2$$

   Repetition continues until the absolute value of $(r_n)^2$ - x is less than some specified threshold value. An easy if not necessarily optimal initial guess is just x. So for example the calculation starts like this for $x = 2$:

   $$\begin{aligned} r_0 &= 2 && (1)\\ r_1 &= (r_0 + 2/r_0)/2 && (2)\\ &= (2 + 2/2)/2 && (3)\\ &= 1.5 && (4)\\ r_2 &= (r_1 + 2/r_1)/2 && (5)\\ &= (1.5 + 2/1.5)/2 && (6)\\ &= 1.1417 \text{ (approximately)} && (7) \end{aligned}$$

   Write a C program that implements this algorithm and compares its results to those obtained with the library function `sqrt()`. Have the program prompt for x, the threshold value, and a maximum number of iterations; do the above-described computation; and print the result, the actual number of iterations, the square root of x as computed using library function `sqrt()`,

and the difference between the value you compute and the one you get from `sqrt()`. Also have the program print an error message if the input is invalid (non-numeric or negative).

Here are some sample executions (assuming you call your program `newton.c` and compile with `make`):

```
[bmassing@dias04]$ ./newton
enter values for input, threshold, maximum iterations
2 .0001 10
square root of 2:
with newton's method (threshold 0.0001):  1.41422 (3 iterations)
using library function:  1.41421
difference:  2.1239e-06

[bmassing@dias04]$ ./newton
enter values for input, threshold, maximum iterations
2 .000001 10
square root of 2:
with newton's method (threshold 1e-06):  1.41421 (4 iterations)
using library function:  1.41421
difference:  1.59472e-12
```

Hints:

- While it may seem from the description of the problem that you'll need a variable for each r_n, in fact you don't; all you need is one that represents the current guess (r_n) and one that represents the previous guess (r_{n-1}).
- You may find the library function `fabs()` useful.

2. (20 points) *NOTE* that the point of this problem is for you to practice using `for` loops, so you must use at least one to get full credit, and I strongly recommend that you do all the needed repetition using `for`.

   Write a C program that gets a positive integer N from the user and prints an $N+2$ by $2N+4$ pattern of stars and spaces like the following:

   For $N = 4$:

```
************
**  ********
****  ******
******  ****
********  **
************
```

   For $N = 7$:

```
**************
**  ************
****  **********
```

```
******  **********
********  ********
**********  ******
************  ****
**************  **
******************
```

Print an error message if what was entered is not a positive integer.

You might find it useful to split your program into several functions, as a way of keeping the main program from being too complicated, and also as a way of not writing similar code over and over. Functions you might find useful in addition to the main one:

- A function that, called with a character `c` and an integer `n`, prints `c` `n` times. Note that you can use `printf` to print a single character, but it's simpler to just use `putchar`.
- A function that, called with three integers (call them `stars1`, `spaces`, and `stars2`), prints a line consisting of `stars1` stars, then `spaces` spaces, then `stars2` spaces.

3 Honor Code Statement

Include the Honor Code pledge or just the word "pledged", plus *at least one of the following* about collaboration and help (as many as apply).[1] Text *in italics* is explanatory or something for you to fill in. For programming assignments, this should go in the body of the e-mail or in a plain-text file `honor-code.txt` (no word-processor files please).

- This assignment is entirely my own work. *(Here, "entirely my own work" means that it's your own work except for anything you got from the assignment itself — some programming assignments include "starter code", for example — or from the course Web site. In particular, for programming assignments you can copy freely from anything on the "sample programs page".)*
- I worked with *names of other students* on this assignment.
- I got help with this assignment from *source of help — ACM tutoring, another student in the course, the instructor, etc. (Here, "help" means significant help, beyond a little assistance with tools or compiler errors.)*
- I got help from *outside source — a book other than the textbook (give title and author), a Web site (give its URL), etc.. (Here too, you only need to mention significant help — you don't need to tell me that you looked up an error message on the Web, but if you found an algorithm or a code sketch, tell me about that.)*
- I provided help to *names of students* on this assignment. *(And here too, you only need to tell me about significant help.)*

[1] Credit where credit is due: I based the wording of this list on a posting to a SIGCSE mailing list. SIGCSE is the ACM's Special Interest Group on CS Education.

4 Essay

Include a brief essay (a sentence or two is fine, though you can write as much as you like) telling me what about the assignment you found interesting, difficult, or otherwise noteworthy. For programming assignments, it should go in the body of the e-mail or in a plain-text file `essay.txt` (no word-processor files please).