Project[3]: Loops

Due Thursday, 9/28/2017, 11:59am

Project Goals

The goals of this project are to:

1. Get you familiar with computing iteration using `while` and `for` loops.
2. Show you how simple it can be to implement complicated-looking functions.

Important Notes:

1. **Formatting:** Make sure that you follow the precise recommendations for the output content and formatting. Your assignment will be auto-graded and any change in formatting will result in a loss in the grade.
2. **Comments:** Header comments are required on all files and recommended for the rest of the program. Points will be deducted if no header comments are included.

Problem 1

It is possible to calculate the square root of a number (for example, 2) by starting with an initial guess and iteratively improving that guess with a simple mathematical operation until the guess is "good enough." In the case of the square root, the rule to use to find square root of a number n is:

```
new_guess <- (old_guess + (n / old_guess)) / 2.0
```

For example, if we want to find the square root of two using this rule, we can start with a guess of 1.0 and repeatedly use the rule in a loop to produce the sequence:

```
1.0, 1.5, 1.416667, 1.414216
```

the last number of which is very close to the square root of 2. The procedure stops when the square of the guess is close enough in absolute value to the number the user entered. How close is close enough? It depends on the application, but for our purposes if the absolute value of the difference between the input and the squared guess is smaller than `1e-5` then we should consider the guess "good enough". You should use `double` data type for all the variables used in your program.

Write a program that asks the user to enter a positive number. Using the procedure described in the previous paragraph, your program should use a loop to compute the square root of the given number. The loop should stop when the absolute value of the difference between the square of the guess and the input number is smaller than `1e-5`. Your program should use a `while` loop or a `do-while` loop.

Your program should print the value of the guess at each iteration. When you print the guess, you should have a minimum of 10 spaces in what you print, and you should print five numbers after the decimal place. For the final answer, you should print five numbers after the decimal place. An example run of the program is:

```
Enter a number: 256
   1.00000
 128.50000
  65.24611
  34.58486
  20.99347
  16.59387
  16.01063
  16.00000
Estimated square root of 256.00000: 16.00000
```

Save your program in a file called `sq_root.c`.

Notes: (1) To compute absolute values, you will need to use the `fabs` function. To use `fabs`, you will need to `#include <math.h>` (2) To read and print `doubles` with `scanf` and `printf`, you should use `%lf`.

Challenge 1 (10 extra credit points): We had you use double-precision floating point numbers for this problem because if you use single-precision floating point numbers then the procedure described above may sometimes fail to find a solution. The way that it fails is that it gets "stuck" repeating the same guess over and over. For the challenge problem, change all of your `double` variables into `float` variables, and add an `int` variable that counts the number of times you have gone through the body of your while loop, starting at zero. Your program should print both the iteration counter and the guess at each step. Your program should be able to detect the "stuck" case and terminate properly. Here's an example:

```
Enter a number: 100
0       1.00000
1       50.50000
2       26.24010
3       15.02553
4       10.84043
5       10.03258
6       10.00005
Estimated square root of 100.00000: 10.00005
```

Save your program in a file called `sq_root_c.c`.

Problem 2

Write a program that asks the user to enter an integer number `n`, and computes the following mathematical series:

$$S = 1^2 - 2^2 + 3^2 - 4^2 + \cdots + (-1)^{n+1} * n^2$$

Your program should use a `for` loop to compute this series. Your program should run as follows (items underlined are to be entered by the user):

```
Enter an integer number: 5
The value of the series is: 15
```

Save your program in a file called `series.c`.

Grading Rubric

Grading will be done for each problem as follows:

Correctly-named file	5%
Header comment	2%
Program compiles	5%
Correctly-reading data from terminal	28%
Correct result printed	60%

Submission details

To submit your project, you will have to save your project files on nomachine:

- create a directory called “project3”
- save your *.c files in that directory
- TO Submit:
  > cd project3
  > submit

The submission script copies all files in the current directory to our directory. You may submit as many times as you like before the deadline, we only keep the last submission.

Academic Honesty

Academic dishonesty is against university as well as the system community standards. Academic dishonesty includes, but is not limited to, the following:

Plagiarism: defined as submitting the language, ideas, thoughts or work of another as one's own; or assisting in the act of plagiarism by allowing one's work to be used in this fashion.

Cheating: defined as (1) obtaining or providing unauthorized information during an examination through verbal, visual or unauthorized use of books, notes, text and other materials; (2) obtaining or providing information concerning all or part of an examination prior to that examination; (3) taking an examination for another student, or arranging for another person to take an exam in one's place; (4) altering or changing test answers after submittal for grading, grades after grades have been awarded, or other academic records once these are official.

Cheating, plagiarism or otherwise obtaining grades under false pretenses" constitute academic dishonesty according to the code of this university. Academic dishonesty will not be tolerated and penalties can include canceling a student's enrollment without a grade, giving an F for the course, or for the assignment. For more details, see the University of Nevada, Reno General Catalog.