Lab 8: Loops (mostly)...

Due date: Monday, February 6, 11:59pm.

Lab Assignment

Assignment Preparation

Lab type. This is an individual lab.

Collaboration. You can talk to other students in the lab, but no code sharing is allowed and no discussions outside the lab period.

Purpose. This is a short lab. It allows you to practice the use of loops.

Programming Style. All submitted C programs must adhere to the programming style described in detail at

http://users.csc.calpoly.edu/~cstaley/General/CStyle.htm

When graded, the programs will be checked for style. Any stylistic violations are subject to a 10% penalty. Significant stylistic violations, epsecially those that make grading harder, may yield stricter penalties. Also note the the Lab 2 requirement for the content of the header comment in each file you submit applies **to each assignment** (lab, programming assignment, homework) in this course.

Testing and Submissions. Any submission that does not compile using the

```
gcc -ansi -Wall -Werror -lm
```

compiler settings will receive an automatic score of 0.

For each program you have to write, you will be provided with instructor's executable and with either a battery of tests if tests are needed. The programs you submit must either produce correct output (if they take no input) or pass all tests made available to you. You can check whether or not a program produces

correct output by running the instructor's executable on the test case, then running yours, and comparing the outputs.

Program Outputs must co-incide. Any deviation in the output is subject to penalties. **PLEASE, USE BINARY EXECUTABLES PROVIDED BY THE INSTRUCTOR!**. The exception is made in case of floating point computations leading to differences in the last few decimal digits.

The Task

You will prepare two types of programs. The first three programs involve producing a number of ASCII drawings using nested `for` loops. The final assignment will have you implementing a number of mathematical computations that involve repetition.

Program 1: Horizontal Bars

Lab 9 will involve you writing programs that produce a variety of graphical images using the PPM file format. To prepare you for this task, the next three programs introduce you to the idea of printing two-dimensional images. All three programs will use ASCII graphics. The first program shall print a series of horizontal bars of different "colors" (characters) as shown below.

HB0. Name the program `horizontal.c`.

HB1. The program shall take no inputs.

HB2. The program shall output three horizontal "bars" rendered as ASCII characters. All three bars shall have the same dimensions: 50 characters long and six (6) characters tall.

HB3. The first bar shall be drawn using the `'#'` character.

HB4. The second bar shall be drawn using the `'.'` character.

HB5. The third bar shall be drawn using the `'@'` character.

The output of your program shall look as shown in Figure 1

The following constraints apply to your code:

HBC1. You must print every character in a single `printf()` statement. Simply writing

```
printf("##################################################\n");
```

is NOT ALLOWED.

HBC2. In addition to `#define`-ing all magic numbers in your program, you shall also `#define` all magic characters.

```
##################################################
##################################################
##################################################
##################################################
##################################################
##################################################
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

Figure 1: Result of running the `horizontal` program.

HBC3. You can either implement this program completely inside the `int main()` function, or you can define a `void` function that prints the required ASCII art.

Program 2: Vertical Bars

The second program in the series asks you to produce three vertical bars as described and shown in the output below.

VB0. Name the program `vertical.c`.

VB1. The program shall take no inputs.

VB2. The program shall output three vertical "bars" rendered as ASCII characters. All three bars shall have the same dimensions: 15 characters long and 15 characters tall.

VB3. The first bar shall be drawn using the `'#'` character.

VB4. The second bar shall be drawn using the `'.'` character.

VB5. The third bar shall be drawn using the `'@'` character.

The output of your program shall look as shown in Figure 2

The following constraints apply to your code:

VBC1. You must print every character in a single `printf()` statement. Simply writing

```
###############.............@@@@@@@@@@@@@@@
###############.............@@@@@@@@@@@@@@@
###############.............@@@@@@@@@@@@@@@
###############.............@@@@@@@@@@@@@@@
###############.............@@@@@@@@@@@@@@@
###############.............@@@@@@@@@@@@@@@
###############.............@@@@@@@@@@@@@@@
###############.............@@@@@@@@@@@@@@@
###############.............@@@@@@@@@@@@@@@
###############.............@@@@@@@@@@@@@@@
###############.............@@@@@@@@@@@@@@@
###############.............@@@@@@@@@@@@@@@
###############.............@@@@@@@@@@@@@@@
###############.............@@@@@@@@@@@@@@@
###############.............@@@@@@@@@@@@@@@
```

Figure 2: Result of running `vertical` program.

```
printf("###############.............@@@@@@@@@@@@@@@\n");
```

is NOT ALLOWED.

VBC2. In addition to `#define`-ing all magic numbers in your program, you shall also `#define` all magic characters.

VBC3. You can either implement this program completely inside the `int main()` function, or you can define a `void` function that prints the required ASCII art.

Program 3: Frames

The third program will print a "frame"-like image shown below.

```
########################################
#......................................#
#......................................#
#......................................#
#......................................#
#......................................#
#......................................#
#......................................#
#......................................#
#......................................#
#......................................#
#......................................#
#......................................#
#......................................#
#......................................#
#......................................#
#......................................#
#......................................#
########################################
```

FR0. Name your program `frame.c`.

FR1. The program will have no inputs.

FR2. The program prints out the "frame" shape shown above. The shape consists of 20 rows and 40 characters in each row.

FR3. The outter part of the frame is formed by the `'#'` character. The outter part is one character thick.

FR4. The inner part of the frame is formed by the `'.'` character.

The constraints are as before:

FRC1. You must print every character in a single `printf()` statement. Simply writing

```
printf("#......................................#\n");
```

is NOT ALLOWED.

FRC2. In addition to `#define`-ing all magic numbers in your program, you shall also `#define` all magic characters.

FRC3. You can either implement this program completely inside the `int main()` function, or you can define a `void` function that prints the required ASCII art.

0.1 Program 4: Taylor/Maclauren Series

The Taylor series is a power series of the form

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \ldots,$$

where $n! = 1 \cdot 2 \cdot \ldots \cdot n$ is the factorial of n, and $f'(a)$, $f''(a)$, $f'''(a)$, etc., are first, second, third, etc. derivatives of some function f at point a.

The Maclauren series is a Taylor series for $a = 0$.

The Taylor and Maclauren series are used to approximate the values of algebraic and trigonometical functions, such as sin(), cos(), log(), etc. In fact, this is how `C`, and many other programming languages compute the values of these functions.

Some functions that can be represented as Taylor (Maclauren) series are:

$$\sqrt{1+x} = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot (2n)!}{(1-2n)(n!)^2(4^n)} x^n = 1 + \frac{1}{2}x - \frac{1}{8}x^2 + \frac{1}{16}x^3 - \frac{5}{128}x^4 + \ldots \text{(for } |x| < 1)$$

$$\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots \text{(for all } x)$$

$$\cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ldots \text{(for all } x\text{)}$$

$$\ln(x+1) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}(x)^n}{n} = x - \frac{x^2}{2} + \frac{x^3}{3} - \ldots \text{(for } |x| < 1\text{)}$$

Create a file `taylor.c` and the matching header file `header.h` containing, respectively, the definitions and the declarations for the following five functions:

```
double sqRoot(double x, int n);
double sine(double x, int n);
double cosine(double x, int n);
double natLog(double x, int n);
int factorial(int n);
```

The first four functions shall compute the value of, respectively, the Taylor series for $\sqrt{x+1}$, $\sin(x)$, $\cos(x)$ and $\ln(x)$ according to the formulas described above. The parameters to these four functions are:

- `x`: the value for which the Taylor series is to be evaluated
- `n`: number of iterations of the Taylor series to compute

E.g., `sqRoot(0.5,3)` shall compute

$$1 + \frac{0.5}{2} - \frac{1}{8} \cdot 0.5^2$$

(notice that 1 is the first iteration of the computation), while `natLog(0.5, 5)` shall compute

$$5 - \frac{5^2}{2} + \frac{5^3}{3} - \frac{5^4}{4} + \frac{5^5}{5}$$

All functions shall check that the parameter n is positive. Otherwise, return 0. `sqRoot()` and `natLog()` shall also check that the input parameter x is in the range of values specified next to the formulas above. If this is not the case, output 0.

The `int factorial(int n)` function takes as input an integer number and returns its factorial $n! = 1 \cdot \ldots \cdot n$ (by definition, $0! = 1$).

Your code for `sqRoot()`, `sine()` and `cosine()` can call `factorial()` as needed.

For each of the four Taylor series functions (`sqRoot`, `sine`, `cosine` and texttt-tnatLog) you are provided with an instructor's test file with a few public test cases. *Please note, that these test cases do not cover all possibile execution paths in your functions.* You may need to design more test cases and test your functions on them. Testing `factorial()` using the `checkit.h` macros is also left up to you.

In your implementation of the functions for this assignment, you are allowed to use arithmetical operations, but **you are not allowed** to use any mathematical functions from `math.h` or `stdlib.h`

Extra Credit

For each of the Taylor series functions, conduct a mini-study to determine the number of iterations necessary to make your Taylor series approximation of the function indistinguishable from the `C`'s computation of the same function from the point of view of `checkit_double()` macro. Prepare a short report describing what you attempted to do, and how you discovered the specific answers. Submit your report as either a plain text file or a .pdf file. Submit any additional code that you created. List all filenames and explain what their purposes are in an appendix to your report.

Submission.

Files to submit. You shall submit **fvie** files:

`horizontal.c`,
`vertical.c`,
`frame.c`, `taylor.h` and `taylor.c`

Files can be submitted one-by-one, or all-at-once.

Submission procedure. Ssh to `unix1`, `unix2`, `unix3` or `unix4`. Submission command:

```
> handin dekhtyar lab08  <your files go here>
```

Testing

Instructor's output for the three drawing programs is provided. For `taylor.c` functions (excluding `factorial()`, a set of public `checkit.h`-style tests is provided as well.