Lab 3: Part 1. Three silly little programs.

Due date: Tuesday, October 6, 11:59pm.

Lab Assignment

Lab 3 is devoted to the study of conditional statements in C. The lab consists of three parts, each with its own deadline and participation rules:

Part	Assignment	Starts	Due	Participation
Part 1	Simple programs	October 5 (Monday)	October 6 (Tuesday)	*Pair programming*
Part 2	IHS if-statement zones	October 7 (Wednesday)	October 12 (Monday)	*Individual*
Part 3	Lab 2 rewrites	October 12 (Monday)	October 16 (Friday)	*Pair programming*

Assignment Preparation

READ THE INSTRUCTIONS FIRST. Please read this document completely **before** you start your work. The submission instructions and the appendix contain a lot of important information about preparing, testing and submitting your programs. In this assignment, you are expected to test your programs using the facilities provided to you by the instructor. Failure to do so may result in errors in your code, and subsequent severe point deductions.

Lab type. Part 1 of the lab is a **pair programming assignment**. The pairs will be designated by the instructor at the beginning of the lab. See the pair programming handout from the first day of classes for more information on what is expected of you.

Collaboration. Students work in pairs, and it is considered cheating, if members of the team (pair) do not work together. Communication between pairs during lab time is allowed, but no direct sharing of code is allowed.

Purpose. The lab allows you to practice the use of conditional statements. It also facilitates learning the use of boolean expressions as conditions in `if` and `switch` statements.

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 a battery of tests. The programs you submit must 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 coincide. Any deviation in the output is subject to penalties (e.g., use of different words in the output). The exception is made in case of floating point computations leading to differences in the last few decimal digits.

Please, make sure you test all your programs prior to submission! Feel free to test your programs on test cases you have invented on your own[1].

The Task

Note: Please consult the instructor if any of the tasks are unclear.

For this lab, you will write and submit three simple programs. The programs will involve standard input/output functions, use of numeric (integer, floating point) variables, and some arithmetic operations in addition to the conditional statements.

Program 1: In or out? (circle.c)

You will write a program that reads from input three numbers: the radius of a circle centered in point (0,0) (the origin of the coordinates), and an x and y coordinates of a point in 2D space.

Your program must check whether the point is inside or outside the circle. Three situations are possible:

- The point is inside the circle. Your program will output the word `"in"`;
- The point is outside the circle. Your program will output the word `"out"`;

[1] In this lab, we provide you with a large collection of test cases. In some future labs, test case development will be a major part of the lab assignment, so it never hurts to start early

- The point is on the circle boundary. Your program will output the word `"on"`;

When writing the program, please do the following:

1. Not other output. Your program will print something **only** once: either `"in"` or `"out"` or `"on"`. There should be NO other `printf()` statements (e.g., asking for input values).
2. Floats. All variables in the program shall be of type `float`.
3. Circles. Recall from high-school math that the equation defining the inside of a circle is
$$x^2 + y^2 < r^2.$$
Similarly, the equation defining the circle boundary is
$$x^2 + y^2 = r^2.$$
In both equations, r is the radius of the circle and (x, y) is a point in 2D Cartesian space.
4. Comparisons. Floating point numbers are hard to compare using C's "==" comparison operator. 49.99999 and 50.00001 may represent the same number, 50, obtained as a result of two different operations (e.g., 100.0/2.0 and 200.0/4.0). Therefore, in order to determine whether your point (x, y) is on the circle boundary, it is not enough to compare $x^2 + y^2$ to r^2.

   Instead, you will introduce a floating point constant `EPSILON` equal to 0.00001 in your program, and will compare the absolute difference between $x^2 + y^2$ and r^2 to it. If said difference is less than `EPSILON` then you consider the point to lie **exactly on the boundary**. You can also use similar comparison operations to test whether the point is inside or outside the circle.
5. circle.c. Name your program `circle.c`.
6. New line. End your `printf()` statements with newline characters.

Example. Here are some sample runs of the program.

```
> more circle-test01
1
0.5
0.5

> circle < circle-test01
in
> more circle-test02
1
0.5
1

> circle <circle-test02
out
> more circle-test03
1
0
```

```
1

> circle <circle-test03
on
> more circle-test04
6.5
3.5
3.5

> circle <circle-test04
in
```

Program 2: Limerick or Haiku? (poetry.c)

Write a program that will read in three integer values, and output either a limerick or a haiku based on the rules outlined below.

Rules. In what follows we refer to the three integers read by your program as `first`, `second` and `third` and assume that `first` was the first integer read, `second` — the second, and `third` — the third.

Among the rules outlined below, the rule with the lowest number always has a precedence. (For example if Rule 1 is ""`first`" is odd" and Rule 2 is "`second` is even", input 3,4,5 should be dealt with according to Rule 1).

Rule 1. If `first + third` is odd, output the **haiku**.

Rule 2. If `first` is greater than or equal to third - second, output the **limerick**.

Rule 3. If `first` is odd or second is even, output the **haiku**.

Rule 4. Output the **limerick**.

Haiku and Limerick. Use the following poetry as output of your program.

Haiku: as the haiku use the following[2]:

```
Its hard to measure
Internet topology
but its rewarding
```

Limerick: as the limerick use the following[3]:

```
A computer found floating in space
Had belonged to a programming ace
Who'd exploited the Earth
For all it was worth,
Destroying the whole human race.
```

Notes. Name your program `poetry.c`. Your program should not have any output other than to print the appropriate piece of poetry.

[2] http://dissertationhaiku.wordpress.com/2009/08/29/computer-science/
[3] http://www.cs4fn.org/limericks/limerickcomp7.php

Examples.

```
> more poetry-test01
1 0 0
> poetry <poetry-test01
It's hard to measure
Internet topology
but it's rewarding
> more poetry-test02
10 3 8
> poetry <poetry-test02
A computer found floating in space
Had belonged to a programming ace
Who'd exploited the Earth
For all it was worth,
Destroying the whole human race.
> more poetry-test03
6 10 26
> poetry <poetry-test03
It's hard to measure
Internet topology
but it's rewarding
> more poetry-test04
6 11 26
> poetry <poetry-test04
A computer found floating in space
Had belonged to a programming ace
Who'd exploited the Earth
For all it was worth,
Destroying the whole human race.
```

Note, input from the `poetry-test01` file matches Rule 1, input from the `poetry-test02` file matches Rule 2 (and does not match Rule 1), input from the `poetry-test03` file matches Rule 3, and input from the `poetry-test04` file matches Rule 4, but does not match previous rules.

Program 3: What's the last digit? (digit.c)

Write a program that reads an integer value from keyboard and outputs the value of its last digit as an English word, as in the following table:

Last digit	Program Output
0	`zero`
1	`one`
2	`two`
3	`three`
4	`four`
5	`five`
6	`six`
7	`seven`
8	`eight`
9	`nine`

Other notes:

No other output. Just as in the two previous programs, this program shall produce no output other than one of the nine words from the table above.

Comparisons. You are looking at exploring an exhaustive list of mutually exclusive possibilities here. Because of this, there is no need in nested comparisons in the program (although they are not prohibited). You are expected to use only `if` statements for your comparisons.

Program name. Name your program file `digit.c`.

Examples.

```
>more digit-test01
41
>digit < digit-test01
one
>more digit-test02
82
>digit < digit-test02
two
>more digit-test03
53
>digit < digit-test03
three
>more digit-test04
4
>digit < digit-test04
four
>more digit-test05
675
>digit < digit-test05
five
>more digit-test06
16
>digit < digit-test06
six
>digit < digit-test07
seven
>more digit-test07
837
>more digit-test08
1008
>digit < digit-test08
eight
>more digit-test09
79
>digit < digit-test09
nine
>more digit-test10
20
>digit < digit-test10
zero
```

Submission.

Who submits. Each pair should submit only one set of files! However, all files must be submitted by the same person (otherwise, it will be very hard for me to collect them all).

Files to submit. You shall submit four files: `circle.c`, `poetry.c`, `digit.c` and `team.txt`.

No other files can be submitted. In fact, if you submit *other* file, or submit one of the three files with an *incorrect filename*, you may receive an email informing you about a submission error, and asking you to resubmit.

team.txt file. Your `team.txt` file shall contain the list of students in your team (pair). For each student, the file shall list the name, and the Cal Poly loginId. E.g., if I were on a team with Clark Turner, we would submit the following `team.txt` file:

```
Alex Dekhtyar, dekhtyar
Clark Turner, csturner
```

Submission procedure. You will be using `handin` program to submit your work. The procedure is as follows:

- `ssh` to `vogon` (`vogon.csc.calpoly.edu`).
- When prompted for `vogon password`, enter your CSL password – the one you use to log onto lab machines in 14-302.
- Upon login, change to your Lab 2 work directory[4].
- Execute the `handin` command. Please note the following:

  ```
  > handin dekhtyar lab03-1 circle.c poetry.c digit.c
  ```

  Please, DO NOT submit binary files.

  `handin` is set to stop accepting submissions 24 hours after the due time.

Grading

The grade for Part 1 of the lab is formed as follows:

`circle.c`	30%
`poetry.c`	40%
`digit.c`	30%

Any submitted program that does not compile earns 0 points.

All programs will be checked for style conformance. Any style violation will be noted. The program will receceive a 10% penalty.

Appendix A. Testing

Test files are available from the Lab 3 `"Tests and Data"` web page. The following is available:

- Instructor's executables. Executable files `circle-alex`, `poetry-alex` and `digit-alex` are available. Please remember to run the

[4] Lab 1 experience should teach you to create a new working directory for each assignment you do for this class inside your `cpe101` directory.

```
chmod u+x <File>
```

command to make these files executable on your system once you download them.

- Tests. A public suite of tests is available for all programs. The filenames are `circle-tests.tar.gz`, `poetry-tests.tar.gz` and `digit-tests.tar.gz`. Please recall, that once you download these files, you need to unpack them using the following two commands (shown on the example of the first file:

```
> gunzip circle-tests.tar.gz
> tar -xvf circle-tests.tar
```

- Test scripts. Test scripts are available for both your program and the instructor's executable. The file names are `circle-tests.csh`, `poetry-tests.csh`, `digit-tests.csh` for your programs, and `circle-alex-tests.csh` `poetry-alex-tests.csh` and `digit-alex-tests.csh` for the instructor's executables.

  Please remember to run the

```
> chmod u+x <File>
```

  command on each of these six files to make them executable on your system.

Please, make sure you run the tests on your programs before you submit! I will be using the Linux `diff` utility to grade your program's output. If `diff` shows the difference in the output of my programs and yours (even if the answers are correct), I will take points off. Your output **must** match my output **exactly**.