Lab 6: Part 1: simple functions . . .

Due date: Friday, February 18.

Note: Lab 6-2 (a pair programming lab) will be assigned to you on Wednesday, so Lab 6-1 will become homework after that.

Lab Assignment

Assignment Preparation

Lab type. This is an **individual lab**.

Purpose. The lab allows you to create and use simple functions.

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.

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. You can check whether or not a program produces correct output by running the `diff` command.

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.

The Task

A small collection of functions: fun.c

Your first assignment is to write a C program that includes a number of simple functions. The functions you will implement are detailed in the table below:

Function	What it does
`float vectorLength(float x, float y, float z)`	length of vector (x, y, z)
`float median(float x, float y, float z)`	the median of `x, y` and `z`
`float mean(float x, float y, float z)`	the mean (average) of three numbers (`x,y,z`)
`int mmDiff(float x, float y, float z)`	determines whether the mean or the meadian is greater
`float pyth(float x, float y, float z)`	computes $x^2 + y^2 - z^2$
`void printVector(float x, float y, float z)`	print vector (`x,y,z`)

Note 1. `mmDiff` function returns the following values:

`1`	if `median(x,y,z)` > `mean(x,y,z)`
`0`	if `median(x,y,z)` = `mean(x,y,z)`
`-1`	if `median(x,y,z)` < `mean(x,y,z)`

Your implementation of `mmDiff` shall call the `mean()` and `median()` functions.

Note 2. `printVector()` shall produce a single line of output. The line shall open with a `"("` followed by the comma-separated values of the components of the vector, and followed by the `")"`. All elements of the vector shall be formatted using the $\%4.2f$ format.

Your program (i.e. `int main()` shall declare three floating point variables (e.g., `x`, `y` and `z`), shall read them one by one from the input, and shall output the following:

1. For vector `(x,y,z)`:
   (a) Output the vector.
   (b) Output the length of the vector.
   (c) Output the median of `x`, `y` and `z`.
   (d) Output the mean of `x`, `y` and `z`.
   (e) Output whether the mean or the meadian of `x`, `y` and `z` is greater.
   (f) Output the difference between $x^2 + y^2$ and z^2.
2. For vector `(x-y, y-z, z-x)`:
   (a) Output the vector.
   (b) Output the length of the vector.
   (c) Output the median of `x-y`, `y-z` and `z-x`.
   (d) Output the mean of `x-y`, `y-z` and `z-x`.
3. For vector `(x+z, y+x, z+y)`:
   (a) Output the vector.
   (b) Output the length of the vector.

(c) Output the median of `x+z`, `y+x` and `z+y`.

(d) Output the mean of `x+z`, `y+x` and `z+y`.

Name your program `fun.c`.

Distances between points: distances.c

Write a program that reads in information about five points in 2D space (ten `float` values), then proceeds to compute two types of pairwise distances for each pair of points. Finally, the program shall output the computed lists of distances. The first distance computed is the Eucledian distance. The second distance is called the Manhattan distance and it is described below.

Your program shall store the points in a two-dimensional array

```
float points[5][2];
```

Your program shall declare, define and use the following functions:

`float distance(float x1, float y1, float x2, float y2):` computes the Eucledean distance in two-dimensional space between two points. The Eucleadian distance is computed as follows:

$$dist((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

`float manhattan(float x1, float y1, float x2, float y2):` computes the Manhatten distance between two points:

$$dist_M((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|.$$

(The origin of the name is relatively straightorward. If the two points represent locations in a city with an East-West and North-South street greed — kind of like Manhattan — then the Manhattan distance tells you how many blocks you have to walk from one point the other.)

Use `fabs()` function from the `math.h` library to compute the absolute value of an expression.

`void printPoint(float x, float y):` prints the coordinates of a point. Use `"%4.2f"` format for each floating point number you print. The coordinates of the point are printed inside parentheses. E.g., `printPoint(1.3, 5.4)` outputs

```
(1.30,5.40)
```

Do NOT output the newline character in this function.

`void printDistance(float x1, float y1, float x2, float y2, float d):` outputs the distance between the two points. `(x1, y1)` and `(x2, y2)` are the points, `d` is the distance (either Eucledean or Manhatten) between the points. This function shall call `printPoint()` to output the two points, and then shall print the distance. The format of the output is shown in the example below.

`printDistance(0.0,0.0, 1.0,1.0, 2.0)` will produce

```
distance: (0.00, 0.00) to (1.00, 1.00) is 2.00
```

This function **shall** output a newline character at the end of the line.

Your program shall output all Eucledean distances first, then it shall output all Manhattan distances. The distances shall be output between all pairs of points in the `points` array, including the distance between each point and itself.

Name your program `distances.c`

Submission.

Files to submit. Submit two files:

`fun.c`,
`distances.c`

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

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

- `ssh` to `vogon` (`vogon.csc.calpoly.edu`).

```
> handin dekhtyar lab06-1  <your files go here>
```

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

Testing and Grading

Any submitted program that does not compile earns 0 points. Please download an run instructor's test to see the exact output produced. Your programs are expected to match this output.