CS16, 10S, Handout to go with H04 (Simple function definitions) (printable PDF)
Available online at: http://www.cs.ucsb.edu/~pconrad/cs16/10S/homework/H04/handout
The assignment is available at http://www.cs.ucsb.edu/~pconrad/cs16/10S/homework/H04

This handout is your reading assignment to go with H04.

- This handout presents a simplified version of a topic that is covered later in the Etter textbook (in Chapter 4)—namely the definition of simple functions.
- We are introducing this topic a bit earlier than in the Etter text, because it is **so fundamental to Computer Science.**
  - Computer Science is the study of algorithms and abstractions
  - A function definition involves both of these key concepts
  - It is, in a sense, an abstraction of an algorithm.
- The material on this handout should be all you need to complete H04.
  - However, if you want to read more about functions—though this is not required for now—you may like to read ahead in Chapter 4, particularly pages 156-160.

Homework H04 involves writing simple function definitions. Here are some example solved problems you can use as models.

Question: Write a function definition for a function called xSquared that takes an integer x as its parameter, and returns an integer value x^2 (or x times x).

Answer:

```
int xSquared(int x)
{
   return x*x;
}
```

Here's some more detail:

- This function definition starts with `int`, because the instructions say it should return an integer value. On this assignment, your function definitions will always start with either `int` (if they are supposed to return a value that is an integer), or `double` (if they are supposed to return a value that is a real number that may have decimals.)
- The next thing that appears is the name of the function, in this case, xSquared
- What comes next is always a set of parentheses. What goes inside the parentheses is a list of parameters.
  - Parameters are special variables that are assigned a value when we call the function (the values that are assigned to these parameters are called arguments.)
  - For example, if we call the function by writing int y= xSquared(3); then 3 is the argument, and x is the parameter that will be assigned the value 3.
  - In this case there is only one parameter, namely x.
  - The variable name x is preceded by the type of x, which is int in this case.
- Then, we have a set of braces { }. These braces mark the beginning and end of the body of the function
- Since this function is a simple computation, we simply have the word return, followed by the formula for what we are trying to compute. On this homework assignment, all the function definitions will take this form.
- We need a semicolon at the end of the return statement, after the expression—and in this case that's the only semicolon we need.

Please turn over for more

Continued from other side

Here's another example:

Question: Write a function definition for a function called xCubed that takes a real number x as its parameter, and returns a real number value x^3 (or x times x times x).

Answer:

```
double xCubed(double x)
{
   return x*x*x;
}
```

The differences here are:

- The function name is different—we have xCubed instead of xSquared.
- The fact that it takes a real number x as its parameter, where x is not necessarily an integer—so in the parentheses `()` we have `(double x)` instead of `(int x)`
- The fact that it returns an answer that is a real number (not necessarily an integer), so we have `double` as the word in front of the name of the function.
- The formula after the word return is different—we multiply x*x*x

One last example—this one has two parameters:

Question: Write a function definition for a function called perimeterOfRectangle that takes real numbers w and h as parameters (which stand for width and height), and returns the perimeter of the rectangle as a real number. The formula is $p = 2w + 2h$

Answer:

```
double perimeterOfRectangle(double w, double h)
{
   return 2*w + 2*h;
}
```

Note here that:

- If we have two parameters, we specify the type in front of each of them, as in `double w, double h`
- In math notation, multplication is implied when we write a number in front of a variable (e.g, $2w + 2h$), or put two single letter variables together as in $a=wh$ (as in area equals width times height). But we cannot do this in most programming languages, including C. We must write the * when we want to multiply, as in `2*w + 2*h`

With those examples, you should be able to complete homework H04.

End of H04 handout