First name (color-in initial)	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	section (9,10,11, 12,1 or 2)	first name initial	last name initial
Last name (color-in initial)	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z			

H04: Due Monday, 10.20 in Lecture

Function Overloading, void functions, call by reference (Ch 4.6,5.1,5.2)
Assigned: Mon 10.13 Total Points: 50

MAY ONLY BE TURNED IN IN THE LECTURE/LAB LISTED ABOVE AS THE DUE DATE,
or offered in person, for in person grading, during instructor or TAs office hours.
See the course syllabus at https://foo.cs.ucsb.edu/16wiki/index.php/F14:Syllabus for more details.

---

(1) (10 pts) Fill in the information below. Also, fill in the A-Z header by

- **coloring in** the first letter of your first and last name (as it appears in Gauchospace),
- writing **either 9,10,11,12,1 or 2** to indicate your **discussion section (lab)** meeting time
- writing your **first and last initial** in large capital letters.

All of this helps us to manage the avalanche of paper that results from the daily homework.

name:	
umail address:	@umail.ucsb.edu

If you collaborated with AT MOST one other person on this homework, write his/her name below. She/he should also have your name on his/her paper.

Reading: *Read Chapter 4, Section 4.6, and Sections 5.1 and 5.2 (If you don't have a copy of the textbook yet, there is one on reserve at the library.)*

Then, answer the following questions. Be sure to check both sides.

2. Section 4.6 discusses Overloading Function Names. Savitch writes that overloading is when you have two or more different functions that both share the same name. Display 4.17 on p. 233 shows a C++ source file that contains two different function definitions, both with the name `ave`.
   a. (5 pts) What lines of that program contain function prototypes for the two version of `ave`?

   b. (5 pts) What lines of that program contain function definitions for the two version of `ave`?

   c. (5 pts) What lines of that program contain function calles for either of the two versions of `ave`?

   d. (5 pts) Explain briefly, in plain english, the difference between the two versions of the `ave` function. Another way of answering the question is: why are two *different* versions of the `ave` function useful or needed? (Answer in whichever way makes more sense to you to explain.)

   e. (5 pts) Explain: HOW DOES THE COMPILER KNOW, for each of the function calls, which version of the `ave` function is being called?

3. (5 pts) Section 5.1 discusses "void" functions in C++. In your own words, what is a "void function"?

4. (10 pts) Section 5.2 introduces "reference parameters", which are parameters where the "actual parameter" passed into the function can be changed by the function, and the change "sticks". Reference parameters create "aliases" for the variables in the caller—e.g. if there is a local variable x in your main function, and you pass it into a function with prototype:

```
1.    void doubleTheValueOf(int &a) {
2.       a = a * 2;
3.    }
```

then when you come back from a function call `doubleTheValueOf(x);` x will have a value that is twice what it was before you called the function.

Write the function definition for a function that uses pass-by-reference to makes an int variable be the absolute value of what it was before the function was called. That is, if the number passed in is non-negative, it does nothing, but if the number is negative, it changes to that number times -1. Your function have the name `makeAbsoluteValue` and should take one parameter called `v` that is an integer, passed by reference.