1. (10 pts) Section 4.4 discusses the “black box” analogy as it relates to functions. Display 4.7 on page 206 shows two different implementations of the function new_balance, and the author says they are "black box equivalent". In your own words (not just copying words out of the textbook), explain what it means for two functions to be "black box equivalent". I really mean it when I say "in your own words". Read the textbook, understand the material, then put the textbook down and write your answer out of the understanding in your HEAD. Don't just copy words from the textbook—copying words is not a useful learning activity.

2. (10 pts) The programming tip on page 208 about nested loops makes the point that if you have nested loops, as in Display 3.15 (which is back in Chapter 3 on pages 160-161), it may be a good place to apply a function. In the example, the explicitly nested loop of Display 3.15 is replaced with an "implied" nested loop by factoring out the code for get_one_total into a separate function. Re-read the nested loop section on pages 160-161, and read the text and code on pages 208-211. You will need this understanding to complete the problem below.

Now, your job. Below are two C++ functions definitions. On the left, boxOfStars is a function that returns a string that when printed, yields a box of stars of width w and height h. On the right lineOfStars is a function that returns a string that when printed, yields a string containing a sequence of stars of length len, without a new line character.
Your job: following the example from the textbook write a new definition of `boxOfStars` that does NOT contain an explicitly nested loop, by using a call to the function definition on the right.

```cpp
string boxOfStars(int w, int h) {
    string result = "";
    for (int i=0; i<h; i++) {
        for (int j=0; j<w; j++) {
            result += "*";
        }
        result += "\n";
    }
    return result;
}
```

```cpp
string lineOfStars(int len) {
    string result = "";
    for (int j=0; j<len; j++) {
        result += "*";
    }
    return result;
}
```

3. (10 pts) Consider the ORIGINAL version of `boxOfStars` that has a nested loop above. Suppose that we made a mistake in the inner loop variable in the `boxOfStars` function, and instead of `for (int j=0; j<width; j++)` we wrote `for (int j=0; j<width; i++)` (note the mistake—incrementing `i` as in Ivan in this loop header instead of `j` as in Jill.) What would happen, and why?
4. (10 pts) Now, again, consider the example boxOfStars above as re-written by you to use the lineOfStars function. Suppose that we changed the loop variable in the lineOfStars function from \( j \) (as in "Jill") to \( i \) (as in "Ivan"). Would there be any problem with that? If so, what? If not, explain why not. Hint: read section 4.5 about "scope", and make reference to "scope" in your answer. The "black box" analogy may also be a useful concept here.

5. 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 \( \text{ave} \).
   a. (5 pts) What lines of that program contain function prototypes for the two version of \( \text{ave} \)?
   b. (5 pts) What lines of that program contain function definitions for the two version of \( \text{ave} \)?
   c. (5 pts) What lines of that program contain function callees for either of the two versions of \( \text{ave} \)?
   d. (5 pts) Explain briefly, in plain english, the difference between the two versions of the \( \text{ave} \) function. Another way of answering the question is: why are two different versions of the \( \text{ave} \) 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?

6. (5 pts) Section 5.1 discusses "void" functions in C++. In your own words, what is a "void function"?

7. (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:

   ```
   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.