1. (5 pts) What line of code (hint: it's a pre-processor directive) do you include at or near the top of a C++ source file if you want to calculate a square root in your program?

2. (4 pts) Fill in the blank:

According to Savitch, a "function invocation" is a fancy way of saying function ________________.

3. (4 pts) Fill in the blank:

According to Savitch, the input to a function, rather than coming from cin, is through its ____________________ ________________________. (two words)
4. There are two ways to do type casting in C++—one that uses the `static_cast<some_type>` notation, and another that uses the `(some_type)` notation, where `some_type` is actually something such as `int`, `double`, etc. Though the book correctly notes that you should only use the `static_cast` syntax in C++ programs, when writing C, you may need the other syntax. (That older syntax is still used in C; the newer C++ `static_cast` syntax is typically NOT available for use in C).

   a. (5 pts) Briefly, in your own words, what does type casting actually mean, i.e. what is it for, or what does it do?

   b. (5 pts) Suppose you have a variable declared as `int count`; and another variable declared as `int sum`. Assume that `sum` and `count` have both been given values, and that you've already checked, `count` is not zero.

   Write a line of code that declares a variable `avg`, of type `double` and assigns it to `sum` divided by `count`, but use a `static_cast` to convert both variables to values of type `double` before the division takes place.

   c. (5 pts) Now write the same line of code, but this time use the older C++ style of type casting (the one that you may have just learned is typically also used in C, even now.)

5. Savitch discusses three concepts that are very important to keep straight, and not confuse: (a) function **declaration** (also called function **prototype**), (b) function definition (c) function **call**. Here is a short C++ program, with line numbers. Please indicate after the program which line number (or range of line numbers, e.g. 3-5 or 7-14) contains the function prototype, function definition, and function call for the `isDivisibleBy` function.
1. #include <iostream>
2. using namespace std;
3. 
4. bool isDivisibleBy(int a, int b);
5. 
6. int main() {
7.     cout << "result for (15,5) is " << isDivisibleBy(5,15) << endl;
8.     cout << "result for (15,5) is " << isDivisibleBy(5,15) << endl;
9. }
10. 
11. bool isDivisibleBy(int a, int b) {
12.     return ( a % b == 0 );
13. }

a. (4 pts) line number(s) or line number range of function prototype (also called "function declaration") for isDivisibleBy

b. (4 pts) line number(s) or line number range of function definition for isDivisibleBy

c. (4 pts) line number(s) or line number range of function calls for isDivisibleBy