1. The first rule of functions is that they should be "small." But how small? Considering the other points in the chapter about functions, suggest some guidelines about you know whether or not a function is too large.

A function should not take up a full screen each function should be transparently obvious and tell a story and each line leads to the next in a compelling order.

A good rule of thumb is no more than 15 lines, and the function will do one thing

1. When you notice identical, or nearly identical, code in your project, it violates the DRY principle. What is the problem with repeated code, and might you do to remove the problem, and how does this make your software easier to change/maintain?

This being in your code can make it more difficult to modify the code if two pieces of code are similar because when you change code. Also with duplication of code your code will be bloated and be harder to read. Another problem with code duplication is to remove the duplication then you will have to track down every time both of the code segment is used and choose which one you will want to use and replace the other one usually fixing any problems that will arise.

1. One kind of "good comment" is when we write comments that document our classes or functions, with the intent that those comments will be transformed into API documentation. What is another situation in our code, where writing a short comment is acceptable?

Warning of consequences: to warn the program of consequences like extremely long run times.

To do comments : tasks that still need to be fleshed out.

Amplification : to draw more attention to really important parts of the code that might seem unimportant.

1. Reflect on the comments that you have written in your own code. If you didn't write a lot of comments, was your code itself descriptive enough? If you did write comments, what was your motivation for writing those comments?

I used to and still comment every line of code if I am confused on what exactly the code does even if it is readable. I did this a lot in algorithms because even if the variable names and functions were written in a way in which you could understand what is happening, rewriting it in plain English would help me better understand each independent step that was happening.

I also write ToDo’s especially when writing the pseudocode so that I keep track of what needs to be done without corrupting the code.