Reading for next time: TCS 3.

A very common problem in programming:
How do you factor out a recurring pattern so that you only have to write it, or (especially!) modify it once?

Follow the Abstraction Principle: Don't do the same thing over and over again. Instead, abstract out the common pattern and give it a meaningful name.

There are several way to apply this principle in C++. The primary one is function definition.

A cardinal rule in C++: You must define something before you use it.

A function does not have to do exactly the same thing each time you use it. It can have parameters which affect its operation. So a parameter is a variable that can have a different value each time you call the function. When you define a function, such as:

```
void drawEdge (double drawTime) {
    robot.forward(1, drawTime);
    robot.turnLeft(1,0.3);
}
```

the parameter drawTime is sometimes called a formal parameter, because we don't know its value when we read the function.

When you call the function, you have to provide an argument or actual parameter corresponding to each formal parameter.

drawEdge(1.5);

In this case case 1.5 is the argument (actual parameter) corresponding to the

formal parameter drawTime.

Having parameters makes your function more versatile. The parameterless drawEdge() would only draw for 1 sec.; the new one will draw for any duration.

When you are designing your functions, you have to think about which parameters are most useful and which are not so useful. You want your functions to be flexible and general, but not too complicated to use.

(C++ does permit you to define functions with optional arguments, so that if you leave some of them off, they get default values. We will not cover this feature in this class. C++ is a very big language. There's lots in it you won't learn about in this class.)

Think of functions as modules that you are putting together to make a larger system. Your modules should have simple, well-defined purposes, usually something that be stated in a simple sentence.

Modularization is one of the primary tools of engineering design. We divide complicated systems into simpler ones so that we can understand them. We want the modules to be small enough so that we can understand them with our limited cognitive abilities.

Generally, most programmers say that a function should not be longer than about a page.

Don't just assume you have got the formula right. Check some special values (including ones you know) to make sure you get the right answer. Of course you can work it out mathematically.

Functions can also return values. The general form of a function definition is:

```
<return type> <function name> ( <parameter declarations> ) {
   <statements>
}
```

In the case of drawEdge, the <return type> was void, which means means the function doesn't return anything, it just does something. In our example, the function name was "drawEdge".

Names (of functions, variables, etc.) in C++ can be any combination of letters, digits, and underscores, beginning with a letter. Upper case and lower case letters are considered different. So these are all different names:

drawEdge, DrawEdge, drawedge, DRAWEDGE, etc.