Adding the Prototypes

When the compiler encounters a function call in your program, it needs information in order to generate the correct code; the compiler doesn't need to know **how** the function is implemented, but it **does** need to know:

- what types each of the **arguments** to the function are (and how many)
- what type of value the function **returns**

That information is provided by a **prototype**, or **function declaration** (as opposed to a function definition).

```
#ifndef DIGITS_H
#define DIGITS_H
int firstDigit(int);
int lastDigit(int);
int numDigits(int);
#endif
```

These prototypes associate the names `firstDigit`, `lastDigit`, and `numDigits` each with a function that **takes** a single `int` as an argument and which **returns** an `int` as its result. These are **function declarations**. **Go ahead and complete the prototypes now.**

In a prototype, **parameter names are optional**. The compiler doesn't care about the names, but they help **you remember** which parameter matches which argument.

```
double focalLength(double d, rouble r1, double r2, double n);
```

Supplying names in a prototype often helps the reader. The parameter names in a prototype are in **prototype scope**; they have no meaning after the prototype ends, and, specifically, they do not need to match the names used in the definition.