Combining toString()

`Student` inherits both `getName()` and `toString()` from `Person`. When you create a **`Student`**, you can use both of those members if they were defined in **`Student`**.

Put that to work by **calling** the **inherited** version of **`toString()`** **from inside** the new overridden **`toString()`** member function. Use the **scope resolution operator** to specify that you wish to call the base class version of **`toString()`**.

```
string Student::toString() const
{
    return Person::toString() // base-class member
        + ", ID: " + to_string(studentID);
}
```

If you forget to use the scope-resolution operator, your program **blows up the stack and crashes**. At least in Java it is polite enough to give you a StackOverflowError when you try to run it. In C++, you'll just see a seg-fault message.

*Don't confuse method **overriding** (which is what we're doing here), with method **overloading**. With overloading, two or more methods have the same name, but different parameter lists. Overloaded methods are in the same class but overridden methods are in a subclass and they must have exactly the same parameters and return type as the method that they are overriding.*