Overriding toString()

When another class (like `Student`) wants to provide a different implementation for a **`virtual`** member function, like **`toString()`**, it must:

1. Use **exactly the same signature** (number and type of parameters) as the original **`virtual`** function in the base class. There are no conversions between **`int`** and **`double`** for instance as with overloading.
2. Return **exactly** the same type as the original member functions.

Let's override **`toString()`** in the **`Student`** class. Here's the header:

```
class Student : public Person
{
public:
    Student(const std::string name, long sid);
    long getID() const;
    std::string toString() const;
private:
    long studentID;
};
```

Note that the prototype is copied **exactly** from **`Person::toString()`**, except for the keyword **`virtual`**. You **do not need to repeat** the word **`virtual`** in the derived class definition, (although you **may** for documentation purposes). A **`virtual`** function is always **`virtual`**, and a non-virtual function **cannot** be made **`virtual`** in one of its subclasses.