Notes for Homework 1

1. Defining the stream insertion operator in the implementation file.

```
// table.h
template <typename T>
class Table {

    template <typename S>
    friend ostream& operator<<(ostream&, const Table<S>&);

    // rest of class definition not shown
}


// in table.cpp
template <typename T>
ostream& operator<<(ostream& out, const Table<T>& t) {

}
```

2. Overloaded function call operator.

```
table(3, 4) = 10;
table.operator()(3, 4) = 10;
```

3. Note that **append_rows** and **append_cols** do not modify the object on which they are called, they return a new table obtained by concatenating the two operands. Basically, any kind of operation on a table is possible through a suitable sequence of append and subtable operations.

4. The addition operator is overloaded as a function template to take an argument of the following type:

```
T (*)(T)
```

Here **T** is the type parameter for the **Table** class. This type is read as *pointer-to-function-that-takes-a-T-value-and-returns-a-T-value*. It a pointer, so the parameter would appear after the pointer operator, like this:

```
T (*f)(T)
```

The idea is to provide an intuitive syntax for updating all of the values in a table according to a given rule. In the first test program, it is used like this:

```
cout << t + square;
```

where **t** is a **Table<int>** and **square** is a stand-alone function that takes an integer and returns the square of that integer. The equivalent function call syntax is:

```
operator<<(cout, t.operator+(square));
```

The expression **t + square** does not modify **t**, it returns a new **Table** containing the squares of values in **t**.

Notice that the **setw** manipulator is used in the test programs. The operator syntax is:

```
cout << setw(5) << t;
```

The equivalent function call syntax is:

```
operator<<(cout.operator<<(setw(5)), t);
```

This has the effect of writing the table values right justified in columns of width 5. Making this work for your **Table** class is a slightly tricky matter. In your overloaded stream insertion operator, the first thing you should do is obtain the width associated to the **ostream** operand. You can do this by calling its **width** method. Then you can call **setw** with the appropriate argument for each table element that is outputted.