Data Structures and Objects
CSIS 3700

Lab 1 — Separate Compilation

Step 1 — Preparation

- Make sure you have updated your copy of the course public repository.
- Copy the lab files into their own separate directory.

Step 2 — Experimentation

- Try to compile each of the files separately. Note the error messages you get.
- Try to compile all of the files simultaneously. Note the error messages you get.

Step 3 — Fixing the errors

- To fix the not-declared-in-scope errors, you'll need to add the prototypes for **`readData()`**, **`sortData()`** and **`printData()`** in **`main.cc`**.
  - The compiler must see a function's prototype before it is called.
- To fix the unresolved reference errors, compile all of the source files simultaneously.
  - Unresolved references are holes that can't be patched

Step 4 — Try out the program

- Run the program. To use it, enter a few numbers, pressing Enter after each one. When done, press Ctrl-D. The numbers you entered should appear in sorted order.

Step 5 — Create a header file

- Create a file named **`lab1.h`**; in this file, place the following lines:

```
1 #include <iostream>
2 using namespace std;
3
4 int readData(int[],int);
5 void sortData(int[],int);
6 void printData(int[],int);
```

- Remove the prototypes in **`main.cc`** and the **`#include`** and **`using`** lines in **`read.cc`** and **`print.cc`**.
- Add the following line at the top of each of the four **`.cc`** files:

```
1 #include "lab1.h"
```

- Recompile and retest the program

Step 6 — Create the Makefile

As pointed out in the slide show, we have not yet realized any advantage to our process. We do that now.

- Create a file named **`Makefile`** and add the following lines:

```
1 lab1: main.o read.o sort.o print.o
2     g++ -o lab1 main.o read.o sort.o print.o
```

NB: Use a Tab to indent the second line. DO NOT USE SPACES!

Translation of the boxed lines: "The file **`lab1`** depends upon the four files **`main.o`**, **`read.o`**, **`sort.o`**, **`print.o`**. To create or update **`lab1`**, run the command **`g++ -o lab1 main.o read.o sort.o print.o`**"

This rule tells the make utility how to create the **`lab1`** file / program. However, it does not have rules for the four **`.o`** files that it must create; add those now.

One of the rules looks like this:

```
1 main.o: main.cc lab1.h
2     g++ -c main.cc
```

Separate the rules with a blank line.

- Add rules for the other three **`.o`** files. They look like the rule above, with the names changed.
- Save the **`Makefile`**, then run the **`make`** command. Note the output.
- Test the program.

Step 7 — Fix the program

There is a bug in the code, as you've seen by now.

- In **`sort.cc`**, change **`i<end`** to **`i<=end`**.
- Run the **`make`** command again. Note the output.
- Test the program.

What to turn in

Turn in a .zip file of the lab 1 folder.