Al Gore's Revenge (AGRev)

Professor Caleb Fowler

June 11, 2019

The parentheses contains the short file name for this homework assignment. Use it for your source file name. This is the name I look for in my scripts.

Problem.

Write a program exploring how the Earth's climate has changed over the past century and projecting what future temperatures might be like. You are going to explore the change in sea levels and the change in air temperature. The world's oceans have risen 1.8 mm over the past century. Forecasts indicate the rate will increase to 3.1 mm/year for the foreseeable future. In addition, the mean July air temperature was measured for the following cities for last year: New York City: 85 °F, Denver: 88 °F, Phoenix: 106 °F, Sacramento: 92 °F. Reports suggest these temperatures will increase by 2 °F over the next 15 years and will continue to increase at that rate into the future. Note: my sample data shown in this assignment may differ from this.

Requirements.

These are the requirements for the assignment. This means they are general and apply to the entire assignment, rather than one specific part. Not every assignment will have a requirements section.

- Display the results of all computations.
- Make your program output as easy to read as possible.
- Your code must conform to the Style Guide (see section below).

Specification Bundles.

You code the specifications from the various groups to control the maximum potential grade you can get for this assignment. The more work you do, the better grade you can get. Specifications are bundled into groups: "A", "B", "C", "D". You must meet the specifications of the lowest group before I will count the specifications for the highest group. Note: sometimes you will get comments stacking on top of each other - that's OK even though you would never do that in the real world.

You want to comment when these specifications begin because I will be looking for them. You want me to find them. To confirm I can find them, use the grep trick discussed below. Check off the boxes when you complete writing the code for that specification - helps keep track of your work.

"D Grade" Specification Bundle.

If you do not complete any specifications or if you forget to add specification comments to your code, this is the grade I will give you. If you do the work - ALWAYS add the comment!

"C" Specification Bundle.

☐ // Specification C1 - Rising Seas
Compute how much the world's oceans rose over the last 100 years.

☐ // Specification C2 - 15 year rise
Compute how much the oceans will rise over the next 15 ...

☐ // Specification C3 - 30 year rise
... and 30 years.

☐ // Specification C4 - 15 year temp
Compute how much the mean July temperatures will increase over the next 15 ...

☐ // Specification C5 - 30 year temp
... and 30 years.

"B" Specification Bundle.

☐ // Specification B1 - Mixed output
Display your output in inches and °C as well as mm and °F.
1 mm = 0.1 cm = 0.03937 inches.

☐ // Specification B2 - Advice
Add an advice section (it's OK if the advice never changes). This advice is for client's running your program.

☐ // Specification B3 - 50 Year Temp
Calculate the temperature for your cities 50 years out.

☐ // Specification B4 - Client Name
Prompt your client for their name. Include the name somewhere in your output (but only use this specification for your input).

☐ // Specification B5 - Constants
Code the conversion factors in Specification B1 as constants with the proper syntax and coding style.

"A" Specification Bundle.

☐ // Specification A1 – Another city
Add another city. You can either research the actual data or make it up.

☐ // Specification A2 – Mean January Temps
Add temperatures for your cities for January as well as July (again you can research these or make it up).

☐ // Specification A3 - 50 Year Sea level Rise
Calculate the sea level rise in 50 years.

☐ // Specification A4 - Prompt for Sea Level Increase
Using a cin statement prompt for the sea level increase instead of using the 3.1mm given as a constant.

☐ // Specification A5 - Headings
Specifications C1, C2 and C4 all ask you to compute different data. Treat each of these as different sections in your output. Given each one it's own header, similar to this:

```
{ blank line }
```

```
HEADER TEXT YOU PROVIDE
=======================
{ Your output goes here.  }
```

Homework Checklist.

Check the following before you turn in your work:

- ☐ You coded your homework.
- ☐ Does it meet all the requirements?
- ☐ Test your code.
  - ☐ Does it compile?
  - ☐ Does it have any compiler warnings?
  - ☐ Does it run?
  - ☐ Does it produce correct output?
  - ☐ Did you use the grep trick to make sure I can see your work?
- ☐ Upload to Canvas.
- ☐ What's the plagiarism checker score?

Sample Run.

Use the same data to check your answers with mine. Figure **??** shows a sample run with sample data.

Due Date.

This assignment is due by 11:59 PM on Sunday on the date on the calendar in Canvas. Example, if this assignment appears on the Canvas calendar during week 2, the assignment will be due that Sunday at 11:59 PM. All the assignments are open the first day of class and you can begin working on them immediately. I encourage you to start sooner rather than later with your homework, these always seem to take longer than you think.

Late Work.

If you miss the due date and time specified above, your work is late. Canvas records the date and time your homework upload COMPLETES. Late work is still acceptable, but it suffers a -15% penalty. You may turn late work in up until MONDAY 11:59 PM AFTER THE ASSIGNMENT WAS DUE. That is, you have 1 day to turn your work in - after that the Canvas drop box closes. Once Canvas closes I will

```
Al Gore's Revenge
Exploration of the changing weather.

WEATHER CALCULATIONS
====================

Calculating changes in sea levels.
----------------------------------
The worlds oceans rose over the last 100 years, by
0.070866 inches or 0.18 centimeters

In 15 years, the oceans will have risen by
1.83071 inches or 4.65 centimeters

In 30 years, the oceans will have risen by
3.66141 inches or 9.3 centimeters


Calculating changes in air temperature.
---------------------------------------
Average July Temperatures, today
New York City:  85
Denver:         88
Phoenix:        106
Sacramento:     92

Average July Temperatures, +15 years
New York City:  110.5
Denver:         114.4
Phoenix:        137.8
Sacramento:     119.6

Average July Temperatures, +15 years
New York City:  136
Denver:         140.8
Phoenix:        169.6
Sacramento:     147.2
```

Figure 1: Sample program run.

not accept an assignment. Do not email your homework files to me; I will not accept them. Keep in mind the time Canvas uses to record your submission - build 5 - 10 minutes into your estimates to upload the file!

Pro-Tip: Get a bare bones copy of your code running and turn it in [1] . Then go ahead and modify it, fix it and whatnot. Upload it with the same name when you finish. That way, if something unexpected happens, you have some working code turned in. Risk management, class, risk management.

[1] If you really want to go pro, get some sort of version control system running (like Git).

How to Turn in your Homework.

Turn homework in by uploading to the appropriate Canvas Dropbox folder. Save your homework as a .cpp file. Don't zip or otherwise compress your files. Create a file with the following naming format: Short_file_name.cpp. Do NOT split your file up into multiple files. I know that is a standard industry practice, but it just get's in the way for this class.

I ONLY accept homework through the Canvas Dropbox. Do not add it to the submission comments or email it to me - I will not accept it. If you are having trouble submitting the assignment, email me immediately. Make sure you upload it a few minutes before the assignment closes in Canvas. If you go over by just one second - you are late.

Style Guide

All programs you write MUST have the following code and/or comments. Again, I look for these elements with my scripts, you want me to find them.

Comments.

Use white space and comments to make your code more readable. I run a program called cloc (count lines of code) which actually looks for this stuff.

End of line comments are only permitted with variable declarations. Full line comments are used everywhere else.

Specification Comments.

Specifications are bundled into groups: "A", "B", "C", "D". You must meet the specifications of the lowest group before I will count the specifications for the highest group. For example, you must meet the "D" specifications before I will count the "C" specifications. If you

miss one element of a specification bundle, that is the grade you will get for the assignment - regardless of how much extra work you do.

Use whole line comments for Specifications. Put the comment on the line above the start of the code implementing the Specification. If the same Specification code appears in more than 1 place, only comment the first place that Specification code appears. Number your Specifications according to the specification bundle and the specific specification you are using, also provide a very short description. DO NOT BUNCH ALL YOUR SPECIFICATIONS AT THE TOP OF THE SOURCE FILE. Example specification comment:

```
// Specification A2 - Display variables
Your code to do this starts here;
```

It's very important to get the specifications down correctly. If your specification code isn't commented, it doesn't count. I use the grep trick to find your specification code. Proper documentation is part of the solution, just like actually coding the solution is.

Compiler Warnings.

Compiler warnings are a potential problem. They are not tolerated in the production environment. In CISP 360 you can have them. I will deduct a small number of points. CISP 400 - I will deduct lots of points if compiler warnings appear. Make sure you compile with -Wall option. This is how you spot them.

C++ Libraries.

We are coding in C++, not C. Therefore, you must use the C++ libraries. The only time you can use the C libraries is if they haven't been ported to C++ (very, very rare).

Non-Standard Language Extensions.

Some compilers support unapproved extensions to the C++ syntax. These extensions are **unacceptable.** Unsupported extensions are compiler specific and non-portable. Do not use them in your programs.

Program Greeting.

Display a program greeting as soon as the program runs. This is a simple description of what the program does. Example:

```
// Program Greeting
cout « "Simple program description text here." « endl;
```

Source File Header.

Start your source file with a program header. This includes the program name, your name, date and this class. I use the grep trick for .cpp (see below) to look for this. I focus on that homework name and display the next 3 lines. Example:

```
// drake.cpp
// Pat Jones, CISP 413
// 12/34/56
```

Specifications and Specification Bundles.

You document specifications like this: // Specification C1 - Some stuff

You do not need to code them in order. You will probably want to because the specifications get harder as you move up in bundles (not THAT much harder). You also don't need to worry about the specification comments appearing in order in your code, either.

However, all of a specification bundle must be coded to reach that bundle grade (ie all C bundle to get a C). Partially completed bundles DO NOT COUNT. Say you code all specifications for a B bundle and only 1 for an A bundle (out of 5 for example). The highest grade you would get would be a B because that's the last bundle you've completed.

You can stop at any bundle you want, you just can't get a higher grade (ex, you code all specifications for bundle B - the best you can get for this homework is a B). This is designed to mirror the work word, the more features your code has, usually, the happier your clients are. This also gives you some control over your grade.

This style guide has more information on the specifics of these comments.

Variables.

Constant variables - anytime you have a value which is not supposed to change, that's a constant. We make it read only with the const keyword and signify it with the ALL CAPS style: const PI = 3.14; We prefer using constants because they make the code easier to read. There are a few situations where we do not usually use them, such as starting a loop at zero. However, if we have that loop end at, say, 33, then it's a magic number. What's 33? Who knows? If we use const SIZE = 33; we know what 33 is.

When we have numeric literals appearing in the program we call these magic numbers. We don;t know what they are, but if we change them, the program breaks. hence, magic. Magic numbers are generally frowned upon.

Grep Trick.

Always run your code immediately before your turn it in. I can't tell you how many times students make 'one small change' and turn in broken code. It kills me whenever I see this. Don't kill me.

You can check to see if I will find your specification and feature comments by executing the following command from the command line. If you see your comments on the terminal, then I will see them. If not, I will NOT see them and you will NOT get credit for them. The following will check to see if you have commented your specifications:

```
grep -i 'specification' homework.cpp
```

This will generate the following output. Notice the specifications are numbered to match the specification number in the assignment. This is what I would expect to see for a 'C' Drake assignment. Note the cd Desktop changes the file location to the desktop - which is where the source file is located.

```
calebfowler@ubuntu:~$ cd Desktop
calebfowler@ubuntu:~/Desktop$ grep -i 'specification' cDrake.cpp
    // Specification C2 - Declare Variables
    // Specification C3 - Separate calculation
    // Specification C1 - Program Output
calebfowler@ubuntu:~/Desktop$
```

This is what I would expect to see for an 'A' level Drake assignment.

```
calebfowler@ubuntu:~/Desktop$ grep -i 'specification' aDrake.cpp
{   // Specification C2 - Declare Variables
    // Specification C3 - Separate calculation
    // Specification B1 - Calculation
    // Specification C1 - Program Output
    // Specification B 2 - double and half
    // Specification A1 - Output Headers
    // Specification A2 - Display variables
calebfowler@ubuntu:~/Desktop$
```

We can also look at the line(s) after the grep statement. I do this to pay attention to code segments.

```
grep -i -C 1 'specification' aDrake.cpp
```

```
calebfowler@ubuntu:~/Desktop$ grep -i -C 1 'specification' aDrake.cpp
int main()
{   // Specification C2 - Declare Variables
    int r_starcreation = 7;              // rate of star creation
--

    // Specification C3 - Separate calculation
    float drake = 0;                     // initialize to 0
    // Specification B1 - Calculation
    drake =  r_starcreation * perc_starswithplanets * ave_numberofplanetslife *
perc_devlife * perc_devintlife * perc_comm *  exp_lifetime;

    // Specification C1 - Program Output
    cout << "The estimated number of potential alien civilizations in the univer
se is ";
--

    // Specification B 2 - double and half
    cout << "Half this value: " << drake * .5 << endl;
--

    // Specification A1 - Output Headers
    cout << endl;
--

    // Specification A2 - Display variables
    cout << "Variables:" << endl;
calebfowler@ubuntu:~/Desktop$
```

We can also use this to look for other sections of your code. The grep command searches for anything withing the single quotes '', and the -i option makes it case insensitive. This is how I will look for your program greeting:

```
calebfowler@ubuntu:~/Desktop$ grep -i -C 1 'greeting' aDrake.cpp

    // Program Greeting
    cout << "This program calculates and displays the number of potential";
calebfowler@ubuntu:~/Desktop$
```

The grep trick is extremely powerful. Use it often, especially right before you turn in your code. This is the best way I can think of for you to be sure you met all the requirements of the assignment.

Client System.

Your code must compile and run on the client's system. That will be Ubuntu Desktop Linux, version 18.04. Remember, sourcefile.cpp is YOUR program's name. I will type the following command to compile your code:

```
g++ -std=c++14 -g -Wall sourcefile.cpp
```

If you do not follow this standard it is likely I will detect errors you miss - and grade accordingly. If you choose to develop on another system there is a high likelihood your program will fail to compile. You have been warned.

Using the Work of Others.

This is an individual assignment, you may use the Internet and your text to research it, but I expect you to work alone. You **may** discuss code and the assignment. Copying code from someone else and turning it in as your own is plagiarism. I also consider isomorphic

homework to be plagiarism. You are ultimately responsible for your homework, regardless of who may have helped you on it.

Canvas has a built in plagiarism detector. You should strive to generate a green color box. If you submit it and the score is too high, delete it, change your code and resubmit. You are still subject to the due date, however. This does not apply if I have already graded your homework.

Often, you will not be able to change the code to lower the score. In this case, include as a comment with your homework, what you did and why you thought it was ineffective in lowering your score. This shows me something very important - you are paying attention to what you are doing and you are mindful of your plagiarism score.

ProTip: Get a bare bones copy of your code running and turn it in. Then go ahead and modify it with bonuses and whatnot. Upload it with the same name so it replaces your previous homework. This way, if something comes up or you can't finish your homework for some reason, you still have something turned in. A "C" is better than a zero. Risk management class, risk management.