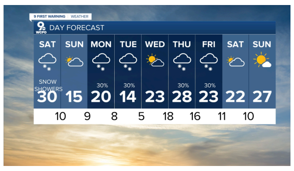
Lab 1 - The Programmer's Almanac

Lab 1 is due **January 22nd, 2024 at 11:59pm**.

Introduction:



Between the hours of 4am and 9am, 4pm and 6pm and 10pm and 12am, it's reasonably likely that if you turn on a **local** **(https://www.wcpo.com/)** **news** **(https://local12.com/)** **broadcast** **(https://www.fox19.com/)** you will find either the anchor or the meteorologist talking about the weather. In a **2018 study** **(https://www.pewresearch.org/journalism/2019/03/26/the-importance-of-local-news-topics-often-does-not-align-with-how-easily-the-public-can-find-information-about-them/)**, 70% of people surveyed by the Pew Research Center identified the weather as one of the local news topics that is important for day-to-day life. If you wound the clock back to 2021 and took a snapshot of what everyone in the country was watching at, say, 12:30pm, you would find that just under **150,000 people were watching the Weather Channel** **(https://www.thewrap.com/weather-channel-fox-news-ratings-battle/)**.

Fox thought that there was enough viewer interest in the weather to launch a competitor to The Weather Channel -- **this past October** **(https://www.adweek.com/tvnewser/inside-fox-weather-on-its-2-year-anniversary/540181/)** marked the two year anniversary since it launched a weather-focused streaming TV service named Fox Weather. Neither The Weather Channel nor Fox Weather is pulling in the ratings of the **NFL or even State of the Union Address** **(https://www.sportico.com/business/media/2024/nfl-posts-93-of-top-100-tv-broadcasts-2023-1234761753/)**, but people who care about the weather, *really* care about the weather.

Even before there was broadcast television to transmit weather reports to our living rooms, people wanted to know about the weather. In fact, the precursor to the National Weather Service was **started right here at the University of Cincinnati** **(https://www.uc.edu/about/factsheet.html)**. And, before the work of Cleveland Abbe (yes, that was the name of the person at UC who pioneered the NWS), Benjamin Franklin worked under a pseudonym to publish **Poor Richard's Almanac** **(https://en.wikipedia.org/wiki/Poor_Richard%27s_Almanack)** as a compendium of data to help farmers predict the weather for the upcoming growing season (it was designed to bring some amusement to families throughout the Colonies, too).

While I am not a weather fanatic like some of my friends, I do care a great deal about living someplace with an ideal climate. (You may, at this point, be asking yourselves why, then, I ended up living in Cincinnati ... we can talk about that at another time). For me, the ideal temperature is somewhere

> between 75F and 85F, with lots of humidity and an almost imperceptible breeze. No fans here, please. I know that not everyone is like me, though.

In this lab you are going to use your newfound C++ knowledge to write an application called *Almanac* that prints a weather forecast that would make *you* smile. So, before you start coding, think about what weather would make you jump out of bed! Would it be a great Fall day? Or a Spring morning? What would the temperature be? Would there be clouds in the sky? How about precipitation? Your ideal forecast is going to contain each of those elements:

1. The season of your ideal day.
2. Whether your ideal day would contain any precipitation.
3. Whether the skies or clear or cloudy, or foggy, or hazy or ... something else.
4. The temperature of your ideal day.

Every lab in this course will have four tasks – a program-design task, a programming task, a critical-thinking task and a question task. In the program-design task, you will be asked to write pseudocode or draw a flowchart for the lab solution you plan to implement. In the programming task, you will be asked to write the C++ code for your solution. In the critical-thinking task, you will be asked to synthesize material you learned in class. In the question task, you will write a question that you had about the material covered in the lab. Every lab document will describe what is required to successfully complete each of the four tasks. At the bottom of each lab document you will find a rubric which will tell you how we plan on scoring your work. I highly recommend reading the *entire* lab document before beginning to work on any part of the solution -- the document usually offers plenty of helpful tips and tricks for getting started.

If you are ever *really* concerned about the weather and need an absolutely certain forecast, don't trust **The Chief** (https://www.fox19.com/authors/steve-horstmeyer/) . Instead, just find a window.

Program-Design Task:

As we learned in class, computer scientists outline their solutions by writing pseudocode. Before you start writing code, please create the pseudocode for your implementation of *Almanac*.

Program-Design Requirements:

Your pseudocode must describe the entirety of the solution. You may choose to write the pseudocode in any way that you like (as a bulleted list, as a flowchart, etc.) and at any level of detail but, remember, this is a tool for you! As we progress through the semester, the programs we write will get more complicated and knowing how to read and write pseudocode (in whatever format that suits you!) will be a tremendous benefit!

Note: Because *Almanac* is a relatively straightforward application, your pseudocode will not need to be very complicated. As stated above, writing pseudocode outlining your solution (*before you start coding*) will be very important in later labs and getting in the habit early is the best way to go! Think of the pseudocode requirement for Lab 1 as simply building muscle memory for the future. No need to overthink it!

Programming Task and Requirements:

Your programming task is to write *Almanac*, an application that prints on the console the forecast of your ideal day. Your program will take no input from the user and the output must meet the following specification (precisely!):

1. The first line of output should introduce the forecast of your perfect day by naming the season in which it would occur. Your choices of season are, of course, Winter, Spring, Summer or Fall. The introduction will be courteous and must start with `My perfect day would occur in the` followed by the season and a colon (`:`).
2. The second line of output should describe your ideal precipitation. The line must start with `Precipitation:` and end with the type of precipitation or *None* if you prefer it dry!
3. The third line of output should describe your ideal skies. The line must start with `Skies:` and end with a description of the skies (e.g., overcast, clear, cloudy, hazy, etc.).
4. The fourth and final line of output should describe your ideal temperature range. The line must start with `Temperature:` and end with a temperature range in Fahrenheit or Celsius.

Note: Your output must contain a space after the `:`s on lines 2, 3 and 4.

For example, the output of my implementation of *Almanac* looks like this:

```
My perfect day would occur in the Spring:
Precipitation: No precipitation
Skies: Clear
Temperature: 75 - 85F
```

Your code must meet the following additional guidelines:

1. To print output to the console, your program must use the objects/methods we learned in class.
2. Your program must use two (2) different methods for outputting a newline character to the console.
3. Your program must contain a multi-line comment before the `main` function describing the purpose of the program. The text in the comment must be grammatically correct and accurately describe the program's purpose, but not its operation.

If you are a Windows-based C++ developer, begin with this **skeleton (https://uc.instructure.com/courses/1657742/files/175921757?wrap=1)** **(https://uc.instructure.com/courses/1657742/files/175921757/download?download_frd=1)** code. If you are a macOS-based developer, begin with this **skeleton (https://uc.instructure.com/courses/1657742/files/175987991?wrap=1)** **(https://uc.instructure.com/courses/1657742/files/175987991/download?download_frd=1)** code. If you are a Linux-based C++ developer, begin with this **skeleton (https://uc.instructure.com/courses/1657742/files/175981411?wrap=1)** **(https://uc.instructure.com/courses/1657742/files/175981411/download?download_frd=1)** code.

Critical-Thinking Task:

Compilation is the process of converting a program written in a *high-level language* into a set of instructions that the CPU can execute. The steps of compilation are "program agnostic" -- in other words, the steps of compilation happen when *compiling* any program and are not specific to that individual piece of code. Your critical-thinking task is to

1. draw a flowchart of the steps of the program-agnostic compilation process, and
2. describe in one sentence each those steps.

Critical-Thinking Requirement:

Your flow chart of program compilation can be written or diagrammatic. If you choose to draw a flow chart, you may draw it by hand. Make sure that all the steps are labeled appropriately. For each step of the compilation process, write one sentence that describes the step. Because there are three steps in the compilation process, you will have three sentences.

Question Task:

For this lab you needed to learn lots of the nuts and bolts about the basics of the C++ language. Along the way you learned some specific new vocabulary, too. Were there any words that threw you off? Were there any pieces of the syntax that didn't make any sense? Please feel free to ask anything related to the first week and a half of class!

Question Requirement:

Submit a file named `question.pdf` which contains your question. Take particular notice of the fact that the question does not have to be subtle, or deep or philosophical for you to receive full points. The goal of the question is for me to be able to get a sense whether there are common questions about the material which would indicate where I failed to teach the material appropriately.

Deliverables:

1. The pseudocode or flowchart describing the algorithm of your *Almanac* application in PDF format (name the file `design.pdf`);
2. The C++ source code for the *Almanac* program that meets all the specifications above (name the file `Almanac.cpp`); and
3. A written or diagrammatic description of the compilation process in PDF format (name the file `compilation.pdf`).
4. A written question about the material covered in this week's lab in PDF format (name the file `question.pdf`).

Submit all four files to Gradescope at the same time. You can access Gradescope from the page for this **Assignment (in Canvas) (https://uc.instructure.com/courses/1657742/assignments/20656592)** or from the left-hand column of links on the course's Canvas page. Gradescope is very, very particular and must be treated with respect. If you do not name your files precisely as defined above (including capitalization and file extensions), Gradescope will get angry. Do *not* assume that Gradescope is satisfied with your submission until you see that you received 40 points!

Remember, you can resubmit as many times as you like until the deadline! Use Gradescope for feedback on how you are progressing.

Tips/Tricks:

To help you get started with Lab 1, feel free to consult versions of Lab 1 that students completed during previous offerings of this very course. You can find them **here (https://uc.instructure.com/courses/1657742/pages/lab-1-menumkr)** and **here (https://uc.instructure.com/courses/1657742/pages/lab-1-in-it-to-win-it)**. Those lab documents contains links to the key -- comparing that lab document with the key with this lab document might give you confidence that you are on the right track!

Rubric:

Your submission will be graded according to the following criteria:

Points	Task	Criteria
10	Design	Pseudocode uses algorithmic thinking to express a high-level description of how to implement the *Almanac* application.
15	Critical Thinking	Description of the compilation process is exhaustive – it contains every step.
15	Critical Thinking	Each step of the compilation process is accurately described in a single sentence.
40	Programming	Program compiles and prints to the console four lines of output meeting the specifications given.
10	Programming	Program uses *two different* methods of outputting a newline character to the console.
5	Programming	`main` function's purpose is described by a grammatically correct multi-line comment.
5	Question	Your question shows an engagement with the material.

Related Learning Objectives:

1. Students will be able to compile and execute a C++ program.
2. Students will be able to use C++ to display output to the console.
3. Students will be able to use comments to document code.
4. Students will be able to write the steps of the compilation process.
5. Students will be able to apply algorithmic thinking to writing pseudocode at varying levels of specificity.