Lab 10 notes:

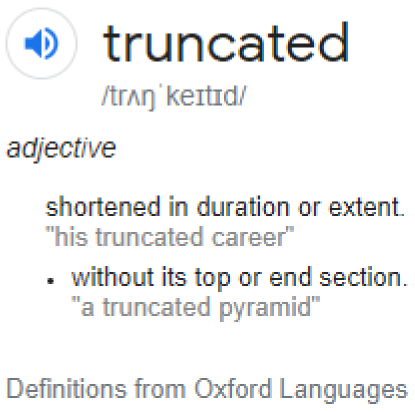
Author: Khang Trinh – 102118468

Part A

Q1: The different types of file open modes are:

Learning Link: https://www.tutorialspoint.com/cplusplus/cpp_files_streams.htm

- app: append mode. All output to the file will be appended at the end
- ate: the file will be opened for output and the read/write control will be from the end of the file
- in: open a file for reading
- out: open a file for writing
- trunc: if the file already exists, its contents will be truncated before opening



truncated
/trʌŋˈkeɪtɪd/

adjective

shortened in duration or extent.
"his truncated career"

- without its top or end section.
  "a truncated pyramid"

Definitions from Oxford Languages

Fig 1. I didn't know what "Truncated" means

Q2: Potential reasons for what would happen if you don't close a file after opening it

Learning Link: https://stackoverflow.com/questions/15854526/why-is-it-necessary-to-close-a-file-after-using-it

- Consume unnecessary system resources
- Your changes to the file may not be saved
- Closing the file = releasing resources. Doing it manually = having control over when/how the programmer wants to release the resources

Q3: A char is 1 byte, an int is 4 bytes and a float is 4 bytes, and the file was 9 bytes. It makes sense!

Learning Link: https://docs.oracle.com/cd/E19253-01/817-6223/chp-typeopexpr-2/index.html

Part B

- When reading, it's worth having "while we're not yet at the end of the file" for more safety precautions
- At first, I thought using `myfile << line << endl;` would be good, but then I found a more efficient example to learn from using `getline()`
- I also found a good example for splitting a string via inputting a delimiter. However, with the current code it only spits out strings, further improvements would need to be made if the output needed to be in other types of data

Part C

- Learned how to import "custom" header files: https://stackoverflow.com/questions/3741051/how-to-include-header-files
- Got my JSON library from the recommended link https://github.com/nlohmann/json/releases
- Learned how a JSON file works https://www.w3schools.com/js/js_json_syntax.asp
- Initial thoughts looking at this: A class with variable names fitting that data set will be required so that I can match the names together to potentially find which data matches which variable to assign them respectively
- Result: Turns out nope! You don't even need a class to match it, I can just treat the json like an object. I wonder how performant it is if the file was to be opened the entire duration of the program and keep reading to it. Maybe assigning the values to variables then close it would be a better idea

-

With this library, you could write:

```
// create an empty structure (null)
json j;

// add a number that is stored as double (note the implicit conversion of j to an object)
j["pi"] = 3.141;

// add a Boolean that is stored as bool
j["happy"] = true;

// add a string that is stored as std::string
j["name"] = "Niels";

// add another null object by passing nullptr
j["nothing"] = nullptr;

// add an object inside the object
j["answer"]["everything"] = 42;

// add an array that is stored as std::vector (using an initializer list)
j["list"] = { 1, 0, 2 };

// add another object (using an initializer list of pairs)
j["object"] = { {"currency", "USD"}, {"value", 42.99} };

// instead, you could also write (which looks very similar to the JSON above)
json j2 = {
  {"pi", 3.141},
  {"happy", true},
  {"name", "Niels"},
  {"nothing", nullptr},
  {"answer", {
    {"everything", 42}
  }},
  {"list", {1, 0, 2}},
  {"object", {
    {"currency", "USD"},
    {"value", 42.99}
  }}
};
```