CS 121 Week 6 - Vectors, Delimiters, and Functions Worksheet

**Introduction:**

Recall the syntax for using ignore with *cin* and *ifstream* objects:

cin.ignore(int, char);

or (assuming infile is an ifstream object that's initialized)

infile.ignore(int, char);

where the int stands for the streamsize skipped and char is the delimiter, a character that marks the end of something, read up to. When calling it, ignore does either one (either read up to streamsize amount of characters or until the delimiter is found). The default values for streamsize and the delimiter are as follows:

//NOTE: input\_stream\_object is something like "cin" or an ifstream object

input\_stream\_object.ignore(1, EOF);

This pretty much means that, if you call "ignore()", you will either skip one character or until you hit the end of the stream/file. You can also call:

input\_stream\_object.ignore(20);

which will attempt to skip twenty characters in whatever input stream you're dealing with, or until it hits the end of your file/stream. Note that the following is syntactically incorrect:

input\_stream\_object.ignore(',');

This is because the function starts with an integer, and cannot accept a character at the beginning. If we wanted to grab a string (i.e. a sequence of characters) until a certain delimiter, we do as follows:

getline(input\_stream\_object, string, char);

where it can be also read as:

getline(input\_stream\_object, string\_to\_save\_to, delimiter);

There are no default values for the first two arguments, however the third (delimiter) on default is '\n' (hence the name getline). Although changing the delimiter makes the function make slightly less sense, know that it's a neat option when working with strings.

Now, using your knowledge of vectors, file IO (input/output), and delimiters, do the following problems. If you are stuck on problem, think about how it should logically work.

**Syntax Practice:**

Assume the console has the following input (... means continuing onwards to an indefinite amount of numbers):

1234.5|77123.3|...

1. Write a code segment that skips the delimiter '|' and reads in each decimal value into a vector. You may create any variables you may need, although (at most) you only need two variables. HINT: You should use a while loop in this code.
2. Assume we have a file named *novel.txt* which consists of the first 500 words in a novel. Omitting commas, semicolons, and anything that does not help in the spelling of a word, each word is separated by a space (with 10 words per line). For reference, each line might look as follows:

The cat ate the hat and sat back on the

couch I do not know why this was the case

...

Create a code segment that opens the file, reads each word in the file and puts them in a vector, then afterwards prints out the number of characters there are in total (i.e. sum up the number of characters per word, then print that sum out). **Do not use getline for this problem.**

**Some Notes (Regarding Functions):**

* A function is a collection of statements to perform a single task. They are meant to make your code easier to read, allow you to type less, improve efficiency, **reusability**, and much more. They enforce the "divide and conquer" programming paradigm, which means to break your program into small pieces until it can be solved directly.
* Functions consist of a return type (either some data type or void for no return), name, parameter list, a body, and require a return statement (if the function is not void type)
* Function prototypes only include the function heading (i.e. the return type, name, and parameter list), ending with a semi-colon
* If you create a function prototype, make sure to create a body later on!
* Function prototype serve to improve readability, and allows anyone reading your code to know what to expect when reading your program
* Parameters are what the function is defined with, whereas arguments are the values actually passed into the function
* When you use a void function, you're saying "I want my function to do this task and give me nothing when it's completed". With any other return type (e.g. int, double, char, etc.) you're saying "I want my function to perform a task and give me a single value back). Best way to think about this is to relate it to math functions.
* The compiler plugs in the body of the function into wherever it finds its function call during compile-time
* Just like before: Make sure your function name relates to what its purpose is, make sure it is concise, and make sure it makes sense! There's a reason why functions like "getline" and "pow" aren't called "goobledyblop" and "whatisthisidonteven".

After reading the above, work on the following problems.

1. Create a function that prints out a user-defined string a user-defined amount of times. The function should take in a string (for what the user wants) and an integer (to account for the amount of times it should be printed). Your function will only print to console and nothing else (i.e. make the return type void).
2. Create a function that prints out a list of numbers from a static array to a file named *num\_output.txt* and returns whether or not it was a successful print. HINT: The function prototype for this to work is the following:

bool did\_print(int num\_list[], int SIZE);

1. Rewrite question 2 (from the first part) so that it is a fully-written program and that it replaces the sum portion into a function. Make sure to include the function call in your program, and have your function prototype look like the following (we'll go over why "const" and "&" are there with the vector later):

int ret\_sum(const vector<string>& list);