# CST126

## Lab 4 – Pulling it all Together

In this lab you will create a program to make a glossary of agile terms. Go here to pick 10 terms for your glossary: <https://www.agilealliance.org/agile101/agile-glossary/>

You can click on any term to see more detail. You need:

* Term Name (Keep your capitalization consistent with these… either initial cap or lowercase all terms.)
* A short one-sentence definition of the term. (You may need to summarize / paraphrase their definition)
* A first used year. (scroll down to origins)

You will have a term struct (or class) to store data about the term and a link to the next term.

You will have a linked list class that:

* Allows you to add terms to your linked list in sorted order.
* Print the names and definitions of your terms in your linked list.
* Allows you to search for a term based.

### Logic

You can hard code 10 terms in the beginning of your main program. DO NOT hard code them in alphabetical order. I want to see that you can add them sorted. (And if I move them around, they should still work. )

Term Class (term.h and term.cpp)

Your term class should have the following member data:

* The name
* The definition (keep these short so that they print nice)
* The start date (an integer)

Your term class will have following member functions:

* A default constructor that blanks/zeros the name, definition and start year.
* A constructor that takes three inputs: one for the name, one for the definition and one for the start year. (You can combine this with the default or not, your choice.)
* A print function that prints out the name and definition.
* A match method that tells you if the name matches an input string.

Linked List (term.h, term.cpp)

You will create a class that contains a list of terms. This class will have a pointer to the top of the linked list as private member data and the following member functions:

* A default constructor.
* A destructor – The destructor should contain a cout statement that shows when it was called. It should delete the linked list.
* A print method that prints the linked list. This function should use the terms print function.
* An add method that adds a term to the linked list. This method must keep the list IN ORDER by term name. See sample flowers code for an example of this. You can use the string comparison operators (<,>) to do this.
* A method that finds and prints a term by name. (Calling the match function in term.)
* Your .h file should have some code in it to make sure it only compiles once. You can use either the #ifndef method or the pragma once method. If you use pragma once, add some comments explaining what it does.

### Output

* Once you’ve created your linked list you should print out the names and definitions. See example below.
* Then, allow your user to enter a term and have your program search your linked list for the practice. Print the term and the first used date like in the sample output.
* Notice that my destructor printed something.
* Align your list output. For example the print the entire list might look like:
* You should pick your own terms and look them up / summarize them. Note that I skipped this step. You don’t get to. 

### Antipattern: Solutions that are counter-productive.

### Burn-down: Definition 8.

### Dates: Include first used dates

### Look-up: You should look up 10

### Refactor: Definition 9.

### Stand-up: Definition 6.

### Story: Definition 5.

### Summary: with summary definitions

### Ten: You should have 10 terms

### Time-box: Definition 7.

### What term do you want? Antipattern

### Antipattern: Solutions that are counter-productive. First used in 1995

### Do you want to continue? n

### Deleting List

### Press any key to continue . . .