# CST126

## Lab 4 – Pulling it all Together

In this lab you will create a program to make a list of conference sessions you want to attend. Find a technical conference that you would like to go to and pick 10 sessions. In the example I did "Agile in the City". You pick your own.

Find a conference you are interested in. You need:

* Session Title (Keep this to under 35 characters. I abbreviated to the main point.)
* Session Speaker. (first and last name)
* A session length in minutes.

You will have a Session **struct** to store data about the Session and a link to the next Session.

You will have a linked list **class** that:

* Allows you to add Sessions to your linked list in sorted order by speaker.
* Print the Title, Speaker and length in minutes for everything in the list.
* Allows you to search for a Session based on the Speaker.

### Logic

You can hard code 10 Sessions in the beginning of your main program. DO NOT add them in alphabetical order. I want to see that you can add them sorted. (And if the grader moves them around, they should still work. ☺). For example I have

session UK1("Descaling agile", "Gojko Adzic", 60);

session UK2("Theory of constraints", "Pawel Kaminski", 90);

and then:

List l;

l.add(&UK1);

l.add(&UK2);

On the output you will see that Gojko ended up in the middle of the list and Pawel at the end even though I added them first.

Session Class (Session.h and Session.cpp)

Your Session struct should have the following member data:

* The Title
* The Speaker
* The session length (an integer)
* A link to the next session.

Your Session **struct** will have following member functions:

* A default constructor that blanks/zeros the title, speaker and session length.
* A constructor that takes three inputs: one for the title, one for the speaker and one for the session length. (You can combine this with the default or not, your choice.)
* A print function that prints out the title, speaker and session length.

Linked List (Session.h, Session.cpp)

You will create a **class** that contains a list of Sessions. 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. It should print the title of each element it deletes.
* A print method that prints the linked list. This function should use the Sessions print function.
* An add method that adds a Session to the linked list. This method most keep the list IN ORDER by Session Speaker. 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 Session by Speaker. Assume that there is only one session per speaker. This function should take advantage of the fact that the list is sorted.
* 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.

### Learning objectives

* Linked Lists and pointers.

### Hint

* You should allow speaker names to be first name and last name. This means that you will need to use cin.getline when you search. Sadly, there is a “quirk” when you combine cin and cin.getline in the same loop. Here is some info about how to solve the problem. <http://www.augustcouncil.com/~tgibson/tutorial/iotips.html#problems>. I had to put an if statement around the ignore.

### Test Data

I will ask the grader to test the following searches:

* The first person on your list.
* A person who lands in the middle of the list.
* The last person on your list.
* A person that isn’t found.

### Output (next page)

* Once you’ve created your linked list you should print out the titles, speakers and session time. See example output.
* Then, allow your user to enter a speaker and have your program search your linked list. Print the Session if you find it or <speaker name > "Not Found" if you don't.
* Notice that my destructor printed something.
* Align your list output. For example, the print the entire list might look like:

### Sessions from Agile in the City for Paula

### Agent of destruction Caryl Regnault 45

### Variety the spice of life Cat Swetel 60

### Governance Cate McLaurin 45

### Liberating Structures David Heath 90

### Descaling agile Gojko Adzic 60

### Clownban Kylie Yearsley 100

### Agile teams, your secret weapon Laura-Jane Parker 45

### It takes two to tango Luca Minudel 45

### Delivering incremental value Maria Scrivener 60

### Theory of constraints Pawel Kaminski 90

### What speaker do you want to find?

### Pawel Kaminski

### Theory of constraints Pawel Kaminski 90

### Do you want to find another? (y/n)n

### -------------------

### ~List

### Agent of destruction

### Variety the spice of life

### Governance

### Liberating Structures

### Descaling agile

### Clownban

### Agile teams, your secret weapon

### It takes two to tango

### Delivering incremental value

### Theory of constraints

### -------------------