

## CS107 Assignment 2: Frequently Asked Questions

1. **Question:** Do we have to use `bsearch`, or can we implement our own search routine?

**Answer:** Our overwhelming preference is that you use the built-in, ANSI C routine called `bsearch`. The `bsearch` function has precisely the same prototype as the `lsearch` function I covered in lecture. It assumes the array elements are laid out to respect the specified comparison function. More information about how `bsearch` works can be found type typing `man bsearch` at the command prompt.

Even though we're less happy with it, we'll allow you to write your own binary search function, but it must be a single search routine that can be called from the implementation of `imdb::getCast` and `imdb::getCredits`. In particular, you should **not** write two, strongly typed binary search routines, because that's code repetition, and code repetition is frowned upon big time, particularly when the code you're replicating is long and algorithmically nontrivial.

Again, we much much prefer you use the library version of `bsearch` instead of writing your own version. The only reason we're allowing this is because the assignment was distributed on Wednesday and we didn't get through the full `lsearch` example until Friday.

2. **Question:** I'm using `bsearch` like you want, but the compiler is complaining when I pass the name of a comparison method as the last argument. What's the problem?

**Answer:** You can't pass a method pointer where a function pointer is expected. Methods (called member functions in CS106B parlance, if I remember correctly) are different from regular functions, because methods expect the address of a receiving object to be passed in via an invisible `this` parameter. As a result, functions and methods aren't interchangeable, so method pointers can't be supplied where a function pointer is expected.

3. **Question:** If I define a top-level comparison function, I don't have access to `actorFile` or `movieFile`. How can I translate integer offsets into movie and actor locations if I don't have access to either of those variables.

**Answer:** Excellent question. The programmer can pass the address of **anything** s/he wants as the first argument of `bsearch`. You're probably passing the address of a string or a film, but it would make more sense to pass the address of a struct as the key. You can include anything you think your comparison function will need to do a full comparison of actors and movies. This requires a genuine understanding of how function pointers and casting works, but it's the genuine understanding that we're shooting for.

4. **Question:** Can I use the `lower_bound` template function from the C++ STL?

**Answer:** Yes, you can if you prefer that over C's `bsearch`, but it'll require you teach yourself material I don't formally cover in CS107 any more. I have a handout on using `lower_bound` and C++ function objects that you're welcome to read. Just email Jerry and ask.

5. **Question:** I think my code is broken: I keep getting that `!Nqate Xqamebe` is the first actor, or it thinks that the second movie connecting Barry to Lou is 2000, not 2001, but that's different from Jerry's handout.

**Answer:** Surprisingly, that guy is an actor, and the handout is wrong on that year. Also, that wasn't a question.

6. **Question:** Whenever I have errors during compilation, the variables in my error messages are displayed as `'a'` instead of the actual variable name.

**Answer:** Add the following line to the file `~/cshrc`:  
`setenv LC_CTYPE C`