Stanford | ENGINEERING
Computer Science

CS 106B: Programming Abstractions in C++ Summer 2014

CS 106B
Home
Textbook
FAQ / Links
Handouts

Course Info
Lectures
Homework
Sections
Exams

Getting Help
Staff/SLs
LaIR Hours
Message Forum
Work@Home

Documentation
Stanford C++ Lib
CppReference.com
CPlusPlus.com
106B Style Guide

Homework 5 (Priority Queue) FAQ

Q: How do I compare strings to see which comes earlier in ABC order?
A: C++ string objects support the standard comparison operators like <, <=, >, >=, ==, and != .

Q: How do I make it so that my `Vector` stores each string and also stores the string's priority? Can I use two vectors?
A: You should have just one vector. Make it a vector of `PQEntry` objects because a `PQEntry` structure stores a string value and an integer priority.

Q: How can I implement `operator <<` for printing a priority queue? It seems like the operator would need access to the private data inside of the priority queue object.
A: The << operator in our assignment is declared with a special keyword called `friend` that makes it so that this operator is able to directly access the private

data inside the priority queue if needed.

Q: What am I supposed to do in a destructor?
A: Free up any dynamic memory that you previously allocated with the `new` keyword.

Q: What am I supposed to do in the VectorPQ destructor?
A: If you don't need to do anything, you can leave something blank.

Q: What is the difference between a destructor and the `clear` method? Don't they do the same thing, deleting all elements from the queue?
A: A clear method is called explicitly when a client wants to wipe the elements and then start over and use the same list to store something else. A destructor is called implicitly by C++ when an object is being thrown away forever; it won't ever be used to store anything else after that. The implementations might be similar, but their external purpose is different.

Q: If I have a vector priority queue storing {"a":1, "b":4, "b":1, "c":3}, and the client calls pq.changePriority("b", 2); , which "b" should I change? "b":4 or "b":1?
A: It should be the first occurrence you find when traversing the vector from left-to-right. So the "b":4 one is the one to change in that case.

Q: What is the difference between `ListNode` and `ListNode*` ? Which one should I use?
A: You literally never want to create a variable of type `ListNode` ; you want only `ListNode*` . The former is an object, the latter is a pointer to an object. You always want pointers to `ListNode` objects in this assignment because objects created with `new` live longer; they are not cleaned up when the current function exits.

Q: How do I declare a `ListNode`?
A: Since the linked list PQ needs to keep its memory around dynamically, you should use the `new` keyword. Like this:

```
ListNode* node = new ListNode();
```

Or, you can pass one or both of the `data` and `next` values on construction:

```
ListNode* node = new ListNode(data, next);
```

Q: For the linked list PQ, how do I make sure that the strings stay in sorted order?
A: You have to do this yourself. You can compare strings to each other using the standard comparison operators: `>= =< > < == !=`

Q: Whenever I try to test my linked list PQ, the program "unexpectedly finishes." Why?
A: This is a very common bug when using pointers. It means you tried to dereference (->) a null pointer or garbage pointer. Run the program in Debug Mode (F5) and find out the exact line number of the error. Then trace through the code, draw pictures, and try to figure out how the pointer on that line could be NULL or garbage. Often it involves something like checking a value like `current->next->data` before checking whether `current` or `current->next` are `NULL`.

Q: How do I resize a heap array when it gets full?
A: Check the `ArrayList` code/slides from lecture to get a good idea.

Q: Is there a difference between "deleting" and "freeing" memory?
A: We use the terms somewhat interchangeably. But what we mean is that you must call 'delete' on your dynamically allocated memory. There is a function named 'free' in C++, but we don't want you to use that.