C++ classes
Due Saturday, February 21, 2015, at 6:00 pm

goal

This project will introduce you to C++ classes and the implementation of quicksort.

where to start

You are to use the template class `array` defined in array.hpp. This file may not be modified.

The implementation of the class should use the code in array.cpp. This file you will need to modify. You may find the sample test driver main.cpp helpful. A sample makefile is given in Makefile.

what you are to do

First, you will need to complete the two constructors for the class

- `template<typename T> array<T>::array(long int N)`
- `template<typename T> array<T>::array(const array<T> &v)`

You will need to initialize the class members `long int n` and `T *a`. The latter you should do with the appropriate call to `new`.

Second, you will need to complete the destructor `template<typename T> array<T>::~array()`. The destructor will need to free the memory allocated in the constructors.

Third, you should complete the implementation of the public member function `quicksort()`. You should implement quicksort so that it sorts the array that `T *a` points to in ascending order.

under the hood

You are free to implement any strategy for choosing the pivot that you wish.

what to turn in, and how

Submit the code in the Project02 Google Drive folder. Place your code for the complete implementation of the `array` class in a file named `array.cpp`. This file should be the ONLY file you submit and must NOT include a `main()`. If I have a test program `main.cpp` like the one provided to you, then your `array.cpp` must be such that

```
g++ -o foo -g -std=c++11 -Wall -pedantic main.cpp array.cpp
```

will successfully build an executable.

grading

No credit will be given for

- code that does not compile and link,
- code that opens any files,
- code that includes an active `main()`.

Here are other problems that will have a negative effect on your grade.

- Your program crashes during testing.
- Code that appears to work but nevertheless has memory access errors.
- Code that is incorrect.
- Memory leaks.
- Code that is unnecessarily inefficient.
- Code that spews output to stdout or stderr. Be sure to disable all debugging i/o!!

tips

- Take a look at the code in the text. It will give you pretty good guidance on the details of what you need to do.
- There are annotations in `main.cpp`, `array.hpp`, and `array.cpp` that indicate where C++ features are discussed in Prata.
- Be sure to test your code on arrays that are large enough to show the runtime (e.g., $n \geq 1,000,000$). If your code is appreciably smaller than your mergesort code on problems of this size, you likely have a bug.
- Be sure to check your code with valgrind! It will detect memory access errors that otherwise might be invisible, and make debugging a LOT easier.
- Be sure that not only is the array sorted upon return, but that it's the same array. A common error is to return values that don't correspond to the original inputs. (I.e., if we return an array of all zeros, it's sorted, but it's probably not the data we wanted to sort.)