CS-102 Design Principles: Project 8

Due date: Friday, May 6 at 11:00pm

This is the final project! You are welcome to come up with your own goals, and you are welcome to collaborate with one other person. Alternatively, you can work on one of the ideas that will be presented later in the instructions.

There is no Github link to accept. To submit your project, you will share your code on Gradescope, along with a video explanation of your project and any other supporting files that you want to include.

You may use any code that we have written together in class or posted on Discord, but not any code taken from anywhere else.

Requirements and options

Whatever idea you go with, your design should use (for a good design reason) at least one class. Your project will be hard-pressed to get a grade above B unless it uses at least one of: inheritance, interface, or an abstract data type (e.g. stack, queue, hash table, priority queue, heap, binary search tree).

Your file structure should adhere to the standards set in the most recent project:

- ☐ Each class has its own `.h` and `.cpp` files; your class declaration should be in the `.h` file, and your method definition(s) should be in the `.cpp` file.
- ☐ Put header guards in the `.h` file, using the convention `FILENAME_H`.
- ☐ Compile all of the `.cpp` files to `.o` object files for each class.
- ☐ Link in the object files when compiling your code.

If you'd like to use the `ncurses` library (that I showed you in class), I can share the instructions for loading it with you. Let me know!

Here are a few ideas for those of you who would prefer not to come up with a completely personal design. You will need to do a bit of research about them:

- ☐ Implement a *skip list*, including all of the usual methods (add, delete, and so on), using linked lists.
- ☐ Implement the *nearest neighbor* heuristic algorithm for the *traveling salesperson problem*.
- ☐ Use a breadth-first search to detect if a graph is *bipartite*.
- ☐ Write the SHA-1 hashing algorithm (or a similar one).

Grading

Your grade will come from:

70%	a working program meeting the guidelines
20%	style
10%	video presentation

Style. Your code must satisfy all of the requirements that we have introduced during this course: Use the conventions that we taught you in class for multi-file projects, use the conventions that we taught you in class for making classes and defining their methods, use `const` appropriately, eliminate dead code, using correct programming constructs, correct indentation, appropriate spacing around parentheses, braces, brackets, and operators, following the variable naming convention and giving reasonable names, and providing file and function headers that are correct and in the prescribed format. Each function in your code should fit in a window (so that they are about twenty lines long at most). Variables should be declared near their initial use. None of your lines of code should exceed 90 characters.

Video presentation. Your presentation should be carefully planned to fit in five to eight minutes. Part of that time should be devoted to showing us what your program does, and the rest should focus on how your project fulfilled the guidelines that we set. Please don't show us your code during your video unless you have an important reason to do so. You can record your presentation in a Zoom session, unless you have another method that you prefer.