CS121 - Computer Science II
Lab Assignment #12
Fall 2011

The purpose of this exercise is to build your own code library.

Over the semester, you have written several functions that perform operations on data structures - in other words, you now have your own "bag of tricks" to do many useful operations on data structures. The purpose of this exercise is to create a library of these functions that can be incorporated in your programs whenever any of the functions are needed.

A library is a special file that includes a table of contents, so that the individual functions can be identified as separate entities. A library of object files, for example, allows the linker to selectively include only those functions that are actually used by your program. By contrast, a regular object file will be included in its entirety, even if some of the functions are never used.

Along with the library itself, usually a header file is written that contains the declarations of the functions and related types included in library. As an example, the standard C library contains all of the functions that are included in the header files, such as `stdlib.h`,`stdio.h`, etc. (One notable exception is the math library, which is in a separate library file and must be explicitly specified in addition to the regular C library).

On Linux/Unix, the name of the program that is used to create library files is `ar`, which stands for "archiver." Actually, it can be used to create a library of any collection of files, but is most often used to create object libraries. An example use of the `ar` command is below:

```
> ar -r  mylib.a objfile1.o objfile2.o
```

In the above example, a library named `mylib.a` will be created, using the functions found in `objfile1.o` and `objfile2.o`. The `-r` indicates **r**eplace, which will insert object files into the library, replacing any current functions with the same name. Other common operations are `-p` (**p**rint), `-d` (**d**elete), and `-x` (e**x**tract). There are other options also; for a more complete list of what `ar` can do, you can type `man ar`.

Once a library is created, it can be specified on the command line for `g++`:

```
g++ main.c subs.c mylib.a
```

In the above example, the compiler will be invoked to compile the files `main.c` and `subs.c`. Then, the library named `mylib.a` will be searched, and any functions that are called from main.c and subs.c will be incorporated into the executable. It is important that the library be specified last, since the linker will only search for functions that have been specified in the previous files. In other words, as the linker processes the modules in `main.c` and `subs.c`, it creates a list of functions that are called, then only searches for those functions within `mylib.a`.

For this assignemnt, you are to create a library of the functions that create and utilize the data structures you have built for the various assignments. You might want to "generalize" these functions, so they can be used in applications other than just the assignments you did for the class. So far, in lab you have created functions that involve:

- Linked lists
- Stacks
- Circular queues
- Queues using linked lists
- The Bubble sort (two versions)
- The selection sort
- The Quicksort (two versions)
- The radix sort.

Note that you do not want to include the `main` programs in your library.

Turn in the file(s) that you use to create your library, along with the header files you wrote to declare the functions in your library. Include all of these files in a zip or tar file, and use `cscheckin` to submit the zip/tar file.