CMSC 106 Project #6 Fall 2010

Due date: Saturday, December 11, 2010

1 Purpose

In this project you will write a program that read records from a file whose name is given through standard input. It will store these records in dynamically allocated arrays, merge arrays, sort arrays and mergeSort arrays.

2 Project description

As you read this section, look at the input files and the source files provided. For this project you are provided with a proj6Main.c file as well as several public testing files.

You must implement each of the functions as described in the `proj6.h` file. You may also add additional functions as helpers for these, but these helpers do not get prototypes in the `proj6.h` file so they can not be called from the files that contain the main functions.

The main creates several empty ItemLists, reads filenames from standard input, open files for reading associated with those names, and then reads from that input file. The input file contains first the size (how many data items are in the file) followed by the data items one on each line.

3 Provided Files

In your `tgz` file, you have:

1. a `proj6.h` file that includes all of the header files you will need. It also defines types that make your implementation easier and prototypes that make your implementation possible.

2. a `proj6.c` file which is a skeleton for you to fill in the code in the required functions. Do not change any of the function headers provided there since they match the corresponding prototypes.

3. a `proj6Main.c` file which is one large run testing many different functions from your `proj6.c`.

4. several `public0?.c` files which each contain a main method designed to test one (or more) of the individual functions you are writing. Each of these has a corresponding `public0?.output` file which shows the output that you should get if your functions are implemented correctly. These files also have their corresponding `data?.in` files which contain file input information in the required format. NOTE: These are NOT to be used with input redirection. They are named inside the file when opening a filepointer.

   You can also write additional test source and input files using these as a model in order to test other functions you write.

4 Project Requirements

If any malloc/calloc or free fails, you must write the exact message "memory allocation error" and exit the program immediately (using the exit command with the value `EXIT_FAILURE` as the argument). All your C programs in this course should be written in ANSI C, which means they must compile and run correctly with `gcc -Wall -Werror -ansi -pedanti` on the grace system as was setup at the beginning of the course. You will lose credit if your program generates any warning messages when it is compiled. Even if you already know what they are, you may not use any C language features other than those introduced so far in lecture or lab. In addition, neither the `goto` nor the `continue` statement may be used, and the `break` statement may not be used in any loop. Using C features not presented in the lectures before the date of the assignment, or using the `goto` statement, or `break` or `continue` in loops **will result** in losing credit. You may use any of the `string.h` functions and any of the `ctype.h` functions already defined in those libraries. Many of these have been discussed in class, and you do not need to use any beyond what has been discussed in class.

IMPORTANT NOTE: If the restriction noted on the implementation of MergeSort is not followed, you will lose all 20 of the style points. You may not merge and then sort.

When doing sorting: If the same value appears more than one time in the list, those values should be placed in the order that they originally appeared into the list from the input file. When doing merge sorting: if the same value appears several times in both lists being merged, all of the items from the first list that contain that specific value should be inserted before any of the values from the list named as the second parameter.

You may not modify the `proj6.h` file at all. All of your code must appear in the `proj5b.c` file. You may add other methods or constants to the `proj6.c` if you would like, but they will not have a prototype in the `proj6.h`.

All work for this project must be your own individual work. You may not work with any other student or compare code in any way. If you need help see either the instructor or TA. Obtaining help from others will be considered cheating. Getting code from a web page or other location will be considered plagerism.

Your program must have a comment near the top which contains your name, login ID, student ID, your section number, and an original description of the action and operation of the program. At the end of this comment block, you must also include the honesty statement: "I pledge on my honor that I have not given or received any unauthorized assistance on this assignment."

Your program should be written using good programming style and formatting, as discussed in class and throughout your textbook. For this project, style is considered to consist of:

- You must implement each of the functions defined in the proj5b.h file without changing the name, return type, parameter(s) or what it does.
- adequate descriptive comments throughout (in addition to the comment mentioned above), to describe what your program is doing and how
- neat and proper indentation and formatting, including consistently and correctly aligning braces using any of the bracing styles illustrated in your textbook
- writing clear and readable code
- use of meaningful descriptive variable names
- use of symbolic constants for all "special" values which don't change and which are used in more than one place in your code
- avoiding disallowed C language features as discussed above
- make sure it runs with the sample input/output files given before submitting it, and make sure you test it on others that are different from those given.