CSE 2320 Lab Assignment 2

Due October 21, 2011

Goals:

1. Understanding of heaps.

2. Understanding of merging.

Requirements:

1. Write a C program to take `n` files containing strings in ascending order (no duplicates within a file) and produce a file `out.dat` containing a line for each string in ascending order. Even if a string `str` appears in multiple files, it should be output *only once* and, for each string, you should also output the number of files (`k`) containing the string. This should be done using code similar to:

   ```
   fprintf(outfp,"%s %d\n",str,k);
   ```

2. Send your program (as an attachment) to `randy.oxentenko@mavs.uta.edu` by 9:45 am on October 21. The `Subject` should be your name as recorded by the University and you should `cc:` yourself to verify that you sent the message correctly. One of the comment lines should indicate the compilation command used on OMEGA.

Getting Started:

1. Your program is to perform only one "heap assisted" merge of all `n` files simultaneously. At any time, there should be no more than one string from each of the input files being processed by your code. It will be useful to have a table of file pointers and a table of strings. Using a heap implementation with "handles" is highly recommended.

   Under no circumstance should your program use multiple binary merges!

2. You may use Sedgewick's heap code (programs 9.11 and 9.12) or code from the course webpage to get started.

3. Your program will be driven by a file `in.dat`:

   a. The first line will contain the value for `n`.
   b. Each of the remaining `n` lines will contain a simple file name, i.e. there will not be a directory path.
   c. Each of the `n` files will contain at least one string. The strings will consist of no more than 50 letters and digits.

4. Pseudo-code:

   a. Open `in.dat`, each of the `n` files, and `out.dat`.
   b. Prime the heap with the first string from each file. The strings will be the priorities, so you will have a minHeap with the smallest (`strcmp()`) string conceptually at the root.
   c. Perform the following processing in each round:
      1. Remove (conceptually) the minimum string from the heap.
      2. if the minimum string is different from the previous minimum
         Output . . .
         else
         Change `k`
      3. Read in another string from the same file as the string just removed.
         if EOF
         heap gets smaller
         else
         Put string in heap
   d. Final clean-up . . .