CSI 402 – Systems Programming – Spring 2014

Programming Assignment II

Date given: Feb. 18, 2014 **Due date:** Feb. 28, 2014
Weightage: 6%

The deadline for this assignment is **11 PM, Friday, Feb. 28, 2014**. With lateness penalty, the program will be accepted until **11 PM, Sunday, Mar. 2, 2014**. The assignment will *not* be accepted after 11 PM on Sunday, Mar. 2, 2014.

Very important: Your source program must consist of two or more C files and you must also have a `makefile`. The C files (with extension "`.c`"), header files (with extension "`.h`") and the `makefile` must be submitted together using the `turnin-csi402` command. Instructions for using `turnin-csi402` and additional specifications for the `makefile` will be included in the `README` file for this assignment.

In this program, you will be converting a text (or formatted) file into a binary (or unformatted) file and vice versa. You will also be carrying out other tasks on binary files.

Each text file will have one or more lines. Each line of such a file has a string (without any intermediate white space characters), followed by a *single tab* character and a non-negative integer. The string has at least one and at most 255 characters. Assume that each integer needs only 4 bytes. For each such text line, the representation in the binary file has one byte which stores the length of the string, followed by the bytes of the string (*without* the terminating `'\0'` character) and the four bytes for the integer. Thus, if a line has a string of length ℓ and an integer, the representation of the line in the binary file will have exactly $\ell + 5$ bytes (one byte for length, ℓ bytes for the string and 4 bytes for the integer). Thus, different lines in the text file may have representations of different lengths in the binary file.

The executable version of your program must be named `p2`. (Your `makefile` should ensure this.) Your program will be executed using a Unix command line of form

`p2` *flag* *infile* *outfile*

or

`p2` `-s` *infile*

Description of the first form: In the first form of the Unix command line, the parameters *infile* and *outfile* represent the names of the input and output files respectively. The parameter *flag* may be either `-t` or `-b`. When the *flag* parameter is `-t`, the input file is a text file and the output file must be the binary file containing the representation for each line as described above. Similarly, when the *flag* parameter is `-b`, the input file is a binary file containing the representations of the lines of a text file, and the output file must be the corresponding text file (with the string and the integer on a line separated by *exactly one tab*).

Description of the second form: In the second form of the Unix command, the command line parameter *infile* represents the name of the *binary* input file (which was created previously from a text file). Note that the only flag that is permitted in this form is `-s`. Here, your program must process the input (binary) file and produce the following information to `stdout`. (In this case, your program should *not* produce an output file.)

(a) The length of a shortest string in the input file.

(b) The length of a longest string in the input file.

(c) The value of the maximum integer in the input file.

(d) The value of the minimum integer in the input file.

You may assume the following. (There is no need to test for these conditions in your program.)

(1) Each text file will have at least one line.

(2) Each string in a text file will have at least 1 and at most 255 characters. Further, the string won't contain white space characters.

(3) Each integer in a text file will be non-negative and can be stored in 4 bytes.

(4) Each binary file will be non-empty; it will contain the correct representation of each line of the corresponding text file.

Your program must detect the following errors. In each case, your program should produce a suitable error message to `stderr` and stop.

(1) Wrong number of command line arguments. (For flags `-t` and `-b`, the correct number of command line arguments is 4; for the `-s` flag, the correct number of command line arguments is 3.)

(2) The specified input or output file cannot be opened.

(3) The flag specified on the command line is not one of `-t`, `-b` or `-s`.

Structural requirement: Your program must must consist of at least two C source files.

Information about `README` file: The `README` file for this assignment will will be available by 10 PM on Saturday, February 22, 2014. The name of the file will be `prog2.README` and it will be in the directory `~csi402/public/prog2` on `itsunix.albany.edu`.