CSE 102 | **Programming Assignment**
W.A. Burkhard | **Due: February 13, 1998**

External Sorting

We are to create an external sorting routine for (fixed sized) records in this exercise. The distribution phase for our routine is will be replacement selection and the merge phase will be the P-way merge algorithm.

The data is the "Current Index to Statistics Extended Database" (CIS) which is a licensed database. The database is line oriented making it rather easy to peruse using a text editor. There is one record per line; the lines do not have the same lengths. Each line begins with a `#` mark and ends with a newline mark. Each field ends with a `#` mark as well. Here is a typical entry designating an article in Advanced Applied Probability:

```
#83AdvAppPr 15 444- 459 J#On the GA/G/$\infty$ queue#Kuczek, Thomas####PV09;94m#
```

There can be other interesting marks within entries; in this instance, the `$\infty$` phrase is Tex for the infinity symbol but is of no significance for us. The pound sign delimiters and newline terminator are the only certain entries on each line. Adjacent pound sign marks signify an empty field. This line is shown here since it will fit easily on this page; many of the lines are considerably longer. The syntax for these records is presented in section 3 of the `USRGUIDE.TXT` file located within the course `public/Data` subdirectory. The third field usually will contain the names of the authors associated with the article. These names will be separated by semicolons.

We will be interested in sorting according to names. Accordingly, we will first construct a summary file that contains fixed length records that contain two values; the anthor's name and the location of the record within the CIS database. There will be one such record for each author associated with an article. Some names in the third field contain extra Tex symbols to designate diacritical marks; in these cases the fifth field is supposed to contain the authors names with all diacritical marks stripped out. Since we will not normally input `P\'olya` but `Polya` if we would like to search for Pólya, we use the stripped name. The third field may also contain a corporate or institutional author such as the `U.S. Census Bureau ...` ; we will not list these articles within our summary file. These entries have a space (mark) in the first position of field three. There is one other feature of authors names – there may be concluded by a role abbreviation. This designates among other possibilities that the person is indeed the author, an editor, a reviewer, or a translator. The role designators are `(Auth)`, `(Ed)`, `(Rev)`, or `(Transl)` respectively. We should remove these as well from our summary; almost everyone would input `Jones, J.R.` rather than `Jones, J.R. (Ed)`. I cannot find the following point within the documentation but it seems that role designators are always preceded by a space. The location is to be the byte offset within the database file of the associated record. We can represent the location either as a string of digits or as a binary number; in the first case, we construct records that are very reasonable to view within an text editor and in the second case, our efforts to construct the record are slightly easier and we save possibly 4 bytes of space per record. Nevertheless I recommend the first approach for ease of viewing.

As a warm up exercise we must create a summary file for our data. The records are to be 64 bytes long. These summary files will be the input to our sorting routine. This portion of the exercise is very short; but it is a necessary step.

Our next step is to implement the replacement selection distribution and the P-way merge methods. Replacement selection is to read our summary file and create a group of initial segments. The average size of these segments should be a (compile-time) parameter of your program; using define statements is a good approach for this. While it would be nice to make these methods perfectly general and work for all sizes of data etc. etc. we will strive for expediency here. Both the replacement selection and P-way merge need only work for your summary files.

The P-way merge method is to combine up to P segments and return one segment. The names of the segments to be combined are provided as inputs to your merge routine.

Both your replacement selection distribution and P-way merge methods should have run-time instrumention. I have placed some routines `102_timer.c` in `public/HW1/Timer` for your use. If you would like to use other (more accurate) tools here, please do so.

What to turnin

0) Groups need only turn in one copy of the work. Your `README` file should contain the names and logins for the partnership. All your files should contain this information at the beginning. The `README` can contain some concluding remarks as well if desired.

1) Your first task is to create a program that will generate the summary files. Let's call this program `summary.c` and the executable `summary`. I suggest that you use the standard input/output library (`stdio.h`); this will result in straightforward code. This program has one input – the name of the data file to summarize.

2) Your second task is to create the external sorting method. This routine/program will have one run-time parameter – the name of the summary file to sort and one compile-time parameter P. Your sort routine must use replacement selection as well as P-way merge. Let's call this executable `sort`. You can turn this in with P equal to 4.

3) You must submit a `makefile` that is capable of creating either of these two executables with commands such as `make sort` or `make summary`.

4) You must submit documentation for both programs. The logic manual which describes how it works is especially important here. The user manual for either will probably be "one liners."

Extra Credit

a. Try different values of P to empirically determine the effect on runtime; $2 \leq P \leq 30$ for instance.

b. Implement an author index with B+trees using the bottom-up construction approach.