CSE 102 **Programming Assignment**
W.A. Burkhard **Due: February 22, 2002**

B+tree Index Creation

We are to create an author index, using the B+tree structure implementation, for data in the "Current Index to Statistics Extended Database" (CIS). The database is line oriented making it rather easy to peruse using a text editor; there is one record per line. Each record contains one bibliographic citation. The lines do not have the same length. The database file is arranged according to journal name; we will be constructing a dense index. The CIS database is a licensed database product; use discretion.

The syntax for the data records is presented in section 3 of the `USRGUIDE.TXT` file located within the course `public/HW1/Data` subdirectory. Our index will be built using author names. Accordingly, we first construct a summary file of fixed length records (64 bytes) that contain two values; the author'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.

There are two programs available to assist us here. The executable `summary` creates a summary file for the data; the records are 64 bytes long. The summary files are then sorted so we can build the B+tree index in a bottom-up fashion. The executable `sort` will sort the summary file created previously.

Our task is to implement the B+tree index facility. We are to create three executables: `makeIndex`, `display` and `access`. The `makeIndex` program is to build the B+tree index structure and store it within the file `<file>.index`; the program has three command-line parameters; the name `-f <file>` of the file containing the sorted summary data, the order `-m <num>` of the B+tree to create, and the number of records `-f <num>` within the external nodes. You will need to include a header record, containing the location of the root, order, etc., within the index file for the `access` and `display` methods. A good place to begin is set the size of both the external and internal nodes to two records/keys.

The `display` program is to show the logical and physical structure of the B+tree; the program has one command-line parameter – the name of the index without its suffix. Details are provided below.

The `access` program has two command-line parameters – an author's name and the index name without its suffix. This program returns a file listing the author's publication records referenced within the index; the file is empty if there are no such records. Details are provided below

Here is a portion of an example output from the `display` program showing the index built for a small, sorted summary file – `sampleData.sort` found in `public/HW1`. The example continues to the top of page 2. The index was built using an order three B+tree and the external nodes contain two records each.

```
B+tree order is 3   external nodes contain 2 records  total number of records 452

logical position    0:   physical position 6528  count is 2      level is 4
     child[0] = 2384
        Harry, Mikel J.
     child[1] = 6380
        Rhee, WanSoo T.
     child[2] = 12448
```

```
logical position    1:   physical position 2384  count is 2      level is 3
     child[0] = 904
        Bradstreet, Thomas E.
     child[1] = 2236
        De Gooijer, Jan G.
     child[2] = 4308

logical position    4:   physical position 904   count is 2      level is 2
     child[0] = 312
        Babu, Gutti Jogesh
     child[1] = 756
        Berman, Simeon M.
     child[2] = 1496

logical position   13:   physical position 312   count is 2      level is 1
     child[0] = 16
        Anderson, Mary R.
     child[1] = 164
        Arratia, Richard
     child[2] = 460

         ...

logical position  115:   physical position 16000     count is 1      level is 0
     child[0] = 28416
        Yokum, Thomas
     child[1] = 28544

logical position  116:   physical position 16444     count is 1      level is 0
     child[0] = 28672
        Zirk, Wayne E.
     child[1] = 28800
```

Here is the file created by the `access` program with input `"Berger, James O."` The pair of records, one per line, are within the `sampleData` file; the lines are broken below to fit on the page.

```
query: Berger, James O.:    2 records found
#92AScintst 80      64-  72 J#Ockham's razor and Bayesian analysis#Jefferys, William
        H.; Berger, James O.####PV18#
#92Biomtrka 79      25-  37 J#Ordered group reference priors with application to the
        multinomial problem#Berger, James O.;Bernardo, Jose M.#Bayesian inference;
        Algorithm###PV18#
```

The `getopt` library function very conveniently reads command-line parameters; a demonstration program is available within HW1.

What to turnin

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

1) Your task is to create programs that will generate, access, and display the B+tree index. Your `makeIndex` program should name the index file `<file>.index`; here `<file>.sort` designates the sorted summary data.

You should use the standard input/output library `stdio.h`; this will result in straightforward code. The `makeIndex` program has three inputs – the name of the sorted summary data file without the `.sort` suffix, the order of the B+tree and the size, that is the number of records, of the external nodes. A first task is to architect the manner in which you will accommodate both internal and external nodes of various and differing sizes. (If you write the program for fixed sized nodes, your task to convert the program allowing for variable sizes will be lengthy.)

2) The `access` program requires two input values – the name of the index (without the `.index` suffix and the author's name; these values are provided via command-line arguments — `-a <author>` and `-f <file>`. The results if any, are to appear in the file named `results`.

3) The `display` program requires one input parameter value – the name of the index provided via a command-line argument `-f <file>` without the `.index` suffix. The output should be directed to the file named `<file>.display`. Your program should logically position nodes as follows: the root is at position 0 and the children of the node in logical position i reside at positions

$$m \times i + 1,\ m \times i + 2,\ \cdots,\ m \times (i + 1).$$

4) You must submit a `makefile` that is capable of creating these three executables with the command `make all`; `make clean` should remove all the executables as well as the dot-oh files.

5) You must submit documentation for your programs. The logic manual which describes how it works is especially important here. The user manual documents client usage for each program — `makeIndex`, `display`, and `access`.

Extra Credit

a. Implement range query access.

b. Implement B+trees using top-down tree construction on unsorted data. How does the time for building an index in this fashion compare with the bottom-up approach we are using.

c. Make a performance study with "random" data counting the numbers of disk accesses for `access` and `makeIndex`.