# CS6360.004 Fall 2016

# Assignment 5

Your assignment is to write a program that implements B+ tree indexing.  It will read a text file containing data and build the index, treating the first 15 columns as the key.  Writing this program will require knowledge of random-access I/O in the language of your choice.  Allowable languages are Java, and C/C++.  
  
The structure of the index is as follows: First 256 bytes: Name of the text file you have indexed.  This must be blank-filled on the right. You may need other metadata in this first block, so I suggest you allocate 1K so you can read it in as a block. (The size of the key is one such piece of metadata; see **1** below.)  The rest of the file is 1K blocks of index data, according to the way a B+ tree is structured.  This implies that your program should read in a block of data, manipulate it, and possibly (if you are inserting key/pointer pairs) write it back out **as a block**.  Use a long (8-byte) record address for your pointers.  These "pointers" are the byte offset in the text file of the data record.  Note that the structure mixes text data (the key) and binary data (the pointer.)  Converting the pointer to text digits is not allowed.  Reading everything into memory, doing the manipulation, and writing it out is also not allowed. Thus you should buffer at most 3 disk blocks.  
  
This program should run entirely from the command line.  Each of the functions and their command-line parameters are given below, assuming the name of your program is INDEX:  
  
**1. Create an index.** This takes three parameters: The name of a text file and the name of an output index that is created, as well as how many byte of the record, starting from the first position, are considered to be the key.  If the output file exists, overwrite it.  Do not ask.  There are two ways to build an index.  The first is to start with a blank structure and insert new keys.  The second is to use a sorted input file of keys and build the tree from left to right.  This second method, while more difficult, is far faster.  However, it is much more difficult to implement, and you need not do it this way.  There may be duplicate keys in the input file.  Your program should list them and the line number of the input file on which they occur.  Duplicates should not be inserted.  The command line is:  
 INDEX -create <input file> <output file> <key size>

For example: INDEX -create CS6360Asg5Data.txt CS6360Asg5.indx 15

Creates an index with a key length of 15 bytes.   
  
**2. Find a record by key.** This displays the entire record, including the key, and gives its position, in bytes, within the file.  If the key is not in your index, the program must give a message to that effect.  As you can guess, the reason for having the name of the text file in the index is so you can find it for this and the subsequent command.  The command line is:  
 INDEX -find <index filename> <key>  
  
for example, if you ran this command line:  
 INDEX -find MyIndex.indx 11111111111111A  
  
your program would return a message that the key was not found.  The command line   
 INDEX -find MyIndex.indx 64541668700164B  
  
would return:  At 2127, record:  64541668700164B ANESTH, BIOPSY OF NOSE  
  (This presumes that the byte offset of this record from the start of the file is 2127).  
  
**3. Insert a new text record.**  As with creating a new index, the first 15 bytes must contain a unique key.  If the key is not in the index, write the record at the end of the data file (remembering the position, since you will need it), then inserting the key into your index structure with the pointer to the start of the new record.  Return a message indicating success and the position of the new record in the text file.  If the key is already in the index, return a message to that effect and do not insert the record.  The program must insert the key into the index and the record into the text file.  The record needs to be in quotes because it could contain spaces and other punctuation.  This has the following command line:  
  INDEX -insert <index filename> "new text line to be inserted."   
  
For example, the following command line would attempt to insert a record into the file.  From the previous example, you know its key is already in your index, so you would get an error message:  
  INDEX -insert MyIndex.indx “64541668700164B Some new Record"  
  
This command line would return success:  
    INDEX -insert MyIndex.indx "11111111111111D Some new Record"  
  
  
**4. List sequential records.**  The program must show the record containing the given key and the next n records following it, if there are that many.  If no record exists with the given key, show records that contain the next-larger key and give a message that the actual key was not found.  As with the -find function, show the records in order by key and their position within the text file.  Obviously this must traverse the tree in key order.  Here is the command line:

  INDEX -list <index filename> <starting key> <count>

**Hint:** It really helps to have a hex file dump utility when you're working on something like this.

Also on eLearning is a script you can use for testing your program. If it runs correctly with the script and does not crash, you’re assured of at least an 80.

**This is a team project.** You are allowed to work with one other person. All work must be done by the team members. If you work with someone else, both names must be in the header comments for each module of your program, and the name of the person who wrote each function must be in header comments for that function.

**To hand in through eLearning:** If you wrote in Java, hand in your source files through eLearning.  If you wrote in C++ using Visual Studio, hand in your entire Visual Studio C++ project.

|  |  |
| --- | --- |
| **Grading Criteria:** | |
| The program works correctly.  That's 15% for each of the sections.  Your program must not crash for bad data. | 60 |
| Good, clean object-oriented design. | 30 |
| Program comments.   An undocumented program will lose all 10 points here.  If you're not sure what good comments are, read my description linked on the main page.  Part of this is using good variable names. | 10 |