

# COP 5536 Spring 2019

## Programming Project

Due Date: Apr 14<sup>th</sup>, 2019, 11:59 pm EST

### 1. General

#### Problem description

The primary value of a B+ tree is in storing data for efficient retrieval in a block-oriented storage context — in particular, file systems. In this project, you're asked to develop and test a small degree B+ tree used for internal-memory dictionaries (i.e. the entire tree resides in main memory). The data is given in the form (key, value) **with no duplicates**, you are required to implement an m-way B+ tree to store the data pairs. Note that in a B+ tree only leaf nodes contain the actual values, and the leaves should be linked into a doubly linked list. Your implementation should support the following operations:

1. Initialize (m): create a new m-way B+ tree
2. Insert (key, value)
3. Search (key): returns the value associated with the key
4. Search (key1, key2): returns values such that in the range  $\text{key1} \leq \text{key} \leq \text{key2}$

#### Programming Environment

You may use either Java or C++ for this project. Your program will be tested using the Java or g++/gcc compiler on the thunder.cise.ufl.edu server. So, you should verify that it compiles and runs as expected on this server, which may be accessed via the Internet.

Your submission must include a makefile that creates an executable file named bplustree.

### 2. Input and Output Requirements

Your program should execute using the following

For C/C++:

```
$ ./bplustree file_name
```

For Java:

```
$ java bplustree file_name
```

Where file\_name is the name of the file that has the input test data.

#### Input Format

The first line in the input file **Initialize(m)** means creating a B+ tree with the order m (note: m may be different depending on input file). Each of the remaining lines specifies a B+ tree operation. The following is an example of an input file:

```
Initialize(3)
Insert(21, 0.3534)
Insert(108, 31.907)
Insert(56089, 3.26)
Insert(234, 121.56)
Insert(4325, -109.23)
Search(234)
Insert(102, 39.56)
Insert(65, -3.95)
Insert(106, -3.91)
Insert(23, 3.55)
Search(23, 99)
Insert(32, 0.02)
Insert(220, 3.55)
Search(65)
```

You can use integer as the type of the key and float/double as the type of the value.

### **Output Format**

For Initialize, Insert and Delete query you should not produce any output.

For a Search query you should output the results on a single line using commas to separate values. The output for each search query should be on a new line. All output should go to a file named “output\_file.txt”. If a search query does not return anything you should output “Null”.

The following is the output file for the above input file:

```
121.56
3.55,-3.95
-3.95
```

## **3. Submission**

Do not use nested directories. All your files must be in the first directory that appears after unzipping.

You must submit the following:

1. Makefile: You must design your makefile such that ‘make’ command compiles the source code and produces executable file. (For java class files that can be run with java command)
2. Source Program: Provide comments.

### 3. REPORT:

- The report should be in PDF format.
- The report should contain your basic info: Name, UFID and UF Email account
- Present function prototypes showing the structure of your programs. Include the structure of your program.

To submit, please compress all your files together using a zip utility and submit to the Canvas system. You should look for Assignment Project for the submission.

Your submission should be named ***LastName\_FirstName.zip***.

Please make sure the name you provided is the same as the same that appears on the Canvas system. Please do not submit directly to a TA. **All email submissions will be ignored without further notification. Please note that the due day is a hard deadline. No late submission will be allowed. Any submission after the deadline will not be accepted.**

## 4. Grading Policy

Grading will be based on the correctness and efficiency of algorithms. Below are some details of the grading policy.

### **Correct implementation and execution: 60%**

Note: Your program will be graded based on the produced output. You must make sure to produce the correct output to get points. Besides the example input/output file in this project description, there are two extra test cases for TAs to test your code. Each one of the test case contributes 20% to the final grade. Your program will not be graded if it can not be compiled or executed.

You will get 0 point in this part if your implementation is not B+ tree.

### **Comments and readability: 15%**

### **Report: 25%**

*You will get 10% points deducted if you do not follow the input/output or submission requirements above. In addition, we may ask you to demonstrate your projects.*

## 5. Miscellaneous

- Do not use complex data structures provided by programming languages. You have to implement B+ tree data structures on your own using primitive data structures such as pointers. You must not use any B tree / B+ tree related libraries.
- Your implementation should be your own. You have to work by yourself for this assignment (discussion is allowed). Your submission will be checked for plagiarism.