Data Structures and Algorithms, CS146, Spring 2018
Programming Assignment II
Due at 11:59pm, on May 1st, 2018

Note:

This programming assignment is for individual work only. You can discuss approaches with other persons, but it is not allowed to use someone's codes or copy codes from Internet. Code plagiarism checker tool will be used to check the similarity of codes. Please check on the University Policies in the syllabus. You may be asked to explain your codes by instructor or to present your codes in the class.

Programming Motivation

The best way to learn data structures and algorithms is to write codes. Writing computer program codes to solve problems using data structures and algorithms is also a required skill for a software engineer working for IT and software industries. Hands-on coding challenges will strengthen your programming skill and experience. In fact, every job interview event/opportunity that you will be encountering in the near future always test your coding kills to see if you can solve problems. If you are a computer science or software engineering major, you **CANNOT** avoid coding, unless you study CS and SE just to kill time for fun and is not for your bright IT career future.
Therefore, enjoy your time in coding and have a lot of fun!!!!

Problem Statements

Based on the simulation of CPU execution queue from the Programming Assignment I, a CPU process might have waiting time in the queue based on the priority code.
For this programming assignment, you are a software engineer and are asked to develop a simulation software application for CPU to **dynamically** keep track of the process scheduling according to the sorted priority codes in the waiting queue. A process with a larger priority code will pre-empt others even if they have been waiting longer in the process queue.

Programming Requirements

1. The CPU process scheduling simulation software application MUST be written in Java and is required to use two separate approaches, **Binary Search Tree** (for sorting) and **Hash Tables** (for searching), which pseudo codes are specified in textbook.
   (You MUST use the pseudo codes provided in textbook to write your Java codes. Any other codes will be treated as failing requirements and **will receive 0 points**.)

2. The CPU process waiting list always contains 20 processes with name of a process and a priority code. Each process will be randomly assigned a distinct priority code.
3. Your simulation software application **MUST** at least contains the following functions:

   (3.1) Process Binary Search Tress (BST):

   a) Build up a Process BST. (**MUST follow BST properties** specified in textbook and ppt slides. Your own tree structure will not be accepted.)
   b) Insert a process to the BST based on its priority code.
   c) Delete a process from the BST.
   d) Make a sorted list of processes according to the priority codes from the BST.

   (3.2) Process Hash Tables:

   a) Create a 11-slots separate **chaining Hash Table** for the 20 processes.
   b) Insert a process to the chaining Hash Table.
   c) Search a process's name by entering a priority code.
   d) Delete a process from the chaining Hash Table.
   e) Make a list of processes in the hash table.
4. Each java file/class/subroutine/function call must contain a header comment in the beginning of it. (Points will be taking off if no comments.)

Submission Requirements

1. The deadline to submit/upload your report and source codes to Canvas is 11:59pm on Tuesday, 5/1/2018. (No Late submission, Please do not wait until the last minute to upload and submit.)
2. Zip all your files into one zipped file with file name: **Your_Last_Name-PA2.zip**, Any file with incorrect file name will not be accepted.
3. You **MUST** write a report (doc or pdf) as much detail as possible with the following items included in the report:

   a) The key concepts of your BST and Hash Table requirement specifications design.

b) Explanations of Classes/subroutines/function calls and of each purpose.

c) Provide sufficient screen shots of each simulation process/procedure including inputs and outputs. This is to verify your codes and to check to see if your codes match the functional requirements.

d) Each test of your program outputs MUST include process's name and priority code. (If testing outputs only show priority code without process's name will receive 0 pints)

e) The procedure (step by step) of how to unzip your files, install your application and run/test your codes.

f) Problems encountered during the implementation.

g) Lesson Learned

4. Grading is based on the full functioning, completeness of requirements and clarity of your codes and report.