Data Structures and Algorithms, CS146-7, Spring, 2018
Programming Assignment 1
Due on March 25, 2018

Note:

This programming assignment is for individual work only. It is not allowed to use someone's codes. Code plagiarism checker tools (Jplag and MOSS) will be used to check the similarity of codes. Please check on the University Policies in the syllabus.

Problem Statements:

A **multitasking operating system** is an operating system that gives you the perception of two or more tasks/jobs/processes running at the same time. It does this by dividing system resources amongst these tasks/jobs/processes and switching between the tasks/jobs/processes while they are executing over and over again. Usually CPU processes only one task/process at a time but the switching is so fast that it looks like CPU is executing multiple processes at a time. This is managed and controlled by a task/process scheduler.
You are a software engineer and are asked to develop a software application to simulate the CPU Process Scheduling System by using Heap Priority Queue. The CPU scheduler can **dynamically** keep track of process scheduling for CPU according to a priority index. A process with larger priority index will pre-empt others even if they have been waiting longer for CPU. If a process has been waiting for a certain period of time (e.g. 50 us), the process will be increased a certain level of priority.

Programming Requirements:

1. The dynamic scheduling software application MUST be written in Java and is required to use Max-Heap Priority Queue approach specified in lecture ppt slides and in textbook. (You MUST directly use the pseudo codes provided in textbook to write your Java codes. Any other codes (e.g. copied from internet) will be treated as failing requirements and **will receive 0 points**.)
2. This Max-Heap priority queue, where elements are prioritized relative to each other and when a process is finished, the software is asked to Extract-Max(S) one process with the highest priority in the queue that is removed.

The priority queue implemented in a Max-Heap Tree will store a collection of process structures that keep track of the process's id (e.g. P1, P2, ...etc.) and an integer index for

the priority. We assume that each process will be randomly assigned a **distinct** priority index integer. Larger integers are considered higher priority than smaller ones (Assuming each time there are more than 20 processes in the MAX-Heap tree.) If a process has been waiting for a certain period of time (e.g. 50 us), the process will be increased a certain level of priority.

3. The software application **MUST** at least contains the following functions:
    A. The following functions must be implemented. (MUST use pseudo codes from the textbook.)
        a) Max-Heapify()
        b) Build-Max-Heap()
        c) Heapsort()
        d) Max-Heap-Insert()
        e) Heap-Extract-Max
        f) Heap-Increase-Key
        g) Heap-Maximum
    B. The software application(AP) must provide a user interface to allow users to insert a process (Your AP will randomly generate a priority index for the process and insert the process to the Heap Tree based on its priority index.)
    C. Allow users to print out the sorted list of all processes including process id and priority index.
    D. Allow users to see and remove the first priority process.
4. Each java file/class/subroutine/function call **MUST** contain a header comment at the beginning of it and each end of line in your codes. . (Points will be taking off if no comments.)

Submission Requirements:

1. The due time to submit/upload your document report, Java source codes and a jar file (so that I can run your codes) to Canvas is before 11:59pm on Sunday, 3/25/2018.
2. Zip all your files into one zipped file with file name: **Your_Last_Name-PA1.zip**, Any file with incorrect file name will not be accepted.

3. You **MUST** write a MS Word report as much detail as possible with the following items included in the report:
    a) Explanations of your system and programming detailed design.
    b) A list of classes/subroutines/function calls and explanations of each purpose.
    c) Provide a lot of 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 and run correctly by yourself. (You will received 0 if no self testing screenshot provided.)
    d) The procedure (step by step) of how to unzip your files, install your application, and run/test your codes.
    e) Problems encountered during the implementation.
    f) Lessons Learned
4. Grading is based on the full functioning, completeness and clarity of your codes and report.