Data Structures and Algorithms, CS146, Spring 2020

Programming Assignment 1:
Airport Control Tower Simulation

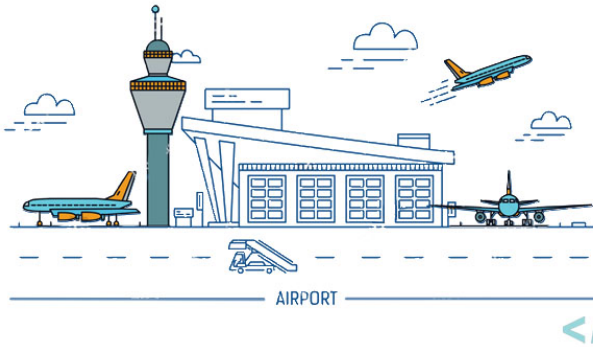
Due at noon, on ~~March 26~~ April 7, 2020
(No late submission! 0 for late submission)

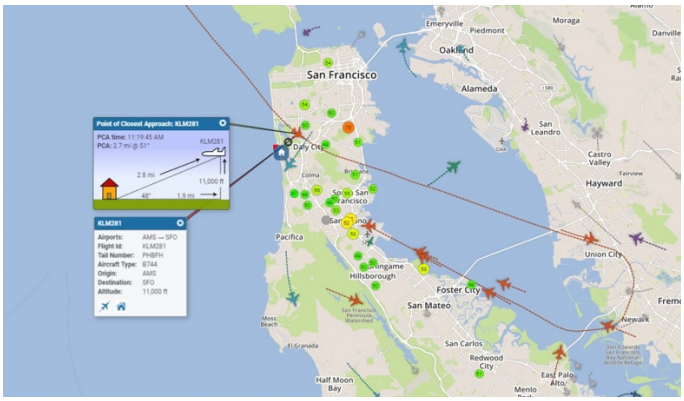
Warning:
This programming **assignment is for individual work only. You are not allowed to use someone's code** or copy from internet. Code plagiarism checker tools (Jplag, MOSS, etc.) will be used to check the similarity of codes. **You will be asked to demo and explain your code to the instructor or to the class. Cheating will not be tolerated** and will be reported to University. Please check on the University Policies in the syllabus. This document is under copyright protected. You are not allowed to upload this document to any websites, otherwise, cheating will be treated (report to SJSU) and your course grade will be changed to F, even after you graduated. (see students caught cheating examples.)

Read this instruction carefully before the design and implementation of your work.

Problem Statements

Air traffic control (**ATC**) is a service provided by ground-based air traffic controllers who direct aircraft on the ground and through controlled airspace. The primary purpose of ATC worldwide is to prevent collisions, organize and expedite the flow of air traffic, and provide information and other support for pilots. Air traffic controllers monitor the location of aircraft in their assigned airspace by radar and communicate with the pilots by radio. To prevent collisions, ATC enforces traffic separation rules, which ensure each aircraft maintains a minimum amount of empty space around it at all times. The primary method of controlling the immediate airport environment is visual observation from the airport control tower. The tower is a tall, windowed structure located on the airport grounds. Air traffic controllers are responsible for the separation and efficient movement of aircraft and vehicles operating on the taxiways and runways of the airport itself, and aircraft in the air near the airport, generally 5 to 10 nautical miles (9 to 18 km) depending on the airport procedures. Air control (known to pilots as "tower" or "tower control") is responsible for the active runway surfaces. Air control clears aircraft **for landing or takeoff**, ensuring that prescribed runway separation will exist at all times. If the air controller detects any unsafe conditions, a landing aircraft may be instructed to "go-around" and be re-sequenced into the landing pattern. This re-sequencing will depend on the type of flight and may be handled by the air controller, approach or terminal area controller.





For this Programming assignment, your job is to design and write java code to simulate the ATC process. For this homework practice purpose, the ATC process has been simplified and is only for airplanes approaching to the airport, but leaving the airport as the following:

1. Each airplane/aircraft approaching has a Flight Number. For example, "KL 445" is a KLM airline service from Amsterdam to Kuwait. A service is called "direct" if it is covered by a single flight number, regardless of the number of stops. For example, "WN 417" flies from Jacksonville to Baltimore to Oakland to Los Angeles. A list of Fight Number can be found at https://en.wikipedia.org/wiki/Flight_number
2. When an airplane is approaching to the airport, the ATC Tower radar will detect it and ATC will receive the airplane information: the flight number, direct distance to runway, and elevation. The information will be used to calculate for its Approach Code.

   Approach Code (AC) = 15000 meters – (Direct Distance to runway in meter + Elevation in meter)/2

Functional Requirements

For this programming assignment, your job is to design and implement an Air Traffic Control Simulator using **Heapsort algorithm and Max-Heap Priority Queue.** The ATC Simulator **MUST** contain the following subroutines specified in textbook:

The following functions must be implemented. (-30% if missing any one subroutine)

1) Max-Heapify()
2) Build-Max-Heap()
3) Heapsort()
4) Max-Heap-Insert()
5) Heap-Extract-Max
6) Heap-Increase-Key
7) Heap-Maximum

In your code, you must make a list of at least 30 airplanes with each one containing a flight number, a direct distance to runway, and an elevation. A random number generator is required to generate a direct distance ranging 3000 - 20,000 meters and another random number generator to generate an elevation ranging 1000 – 3000 meters for an airplane. Approach Code is assigned to each airplane based on the equation below:

Approach Code = 15000 meters - (Direct Distance to runway in meter + Elevation in meter)/2

Two major features of ATC Simulator must be implemented:

1. Using Heapsort algorithm to sort and display the list of airplanes based on the calculation of their Approach Codes (AC) above.

a) Allow users to **add** an airplane's by entering a Flight Number and the other two numbers are randomly generated (-20 points if hardcoded or not functioning.)

-- Example: entering AA20, your application records (AA20, D:10020 meters, H:1005 meter)

b) Calculate AC for each airplane.

c) Store the AC to each airplane.

d) Use Heapsort algorithm to sort at least 30 ACs. **(-20 if less than 30 ACs)**

e) An airplane with a higher Approach Code has a higher priority to land the runway.

f) Allow users to display the sorting result containing at least 30 airplanes in the following output format: (-100 points if hardcoded or -50 points if not functioning.)

-- Example Output: **(-30 if not in this format)**

Airplanes Landing sequence:

1. (US120, D:8500 meters, H: 1000 meters) – AC: 10250
2. (CI06, D:9500 meters, H:1100 meters) – AC: 9700
3. (AA20, D:10020 meters, H:1050 meters) – AC: 9465
4. …etc.

2. Using Heap Priority Queue to store airplanes based on ACs.

a) Implement Heap Priority Queue approach to store the sorted 30 ACs airplanes into Heap (you may use Array, ArrayList or Vector in Java) (-50 points if not functioning.)

b) Allow users to insert a new airplane into Heap based on its AC by implementing Max-Heap-Insert() (-20 points if not functioning.)

c) Allow users to view the first ranked AC airplane (The first one in the queue to land) by implementing Heap-Extract-Max(). (-20 points if not functioning.)

d) In case of emergency landing, allow users to choose **any one** of the airplanes stored in the Heap Priority Queue and increase its AC score with its priority in the queue to be listed by implementing Heap-Increase-Key(). (-20 points if not functioning.)

Programming Requirements

1. The Air Traffic Control Simulator MUST be written in Java and is required to use Heapsort and Max-Heap Priority Queue approach specified in lecture ppt slides and in textbook. (You MUST use and revise the pseudo codes provided in textbook to write your Java codes. Any other codes will be treated as failing requirements and **will receive 0 point**.)
2. Each java file/class/subroutine/function call **MUST** contain a header comment at the beginning of each and make sure you have enough comments at the end of lines in your code. (-20 points if your code did not provide enough comments.)
3. A professional report (clean and organized) is required in IT industry. You **MUST** write a **professional/formal** report in MS Word/PDF format with the following items included in the report: **(You will receive 0 point if a report is not submitted.)**
   a) Cover Page – HW1 Title and your name **(-10% if missing)**
   b) Design and Implementation: **(-20% if missing)**
      Explain/illustrate how you design and implement your Air Traffic Control Simulator such as diagram, graph, data structure, (Comparable Object, Array, ArrayList, Linked List, Vector, etc.)
   c) A list of classes/subroutines/function calls: **(-20% if missing)**
      Explain purpose for each.
   d) Self-testing screen shots:

Provide enough screen shots for each function including inputs and outputs for each of function listed in "Functional Requirements" section. 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 receive 0 point if no self-testing screenshots provided. Points will be taken off if not enough self-testing screen shots provided.)

e) Show and explain your procedure (step by step) of how to unzip your files, install your application, and run/test your codes. (-20 points if not provided)

f) Problems encountered during the implementation: Describe all problems found/encountered during your implementation and how you resolved them. (-20% if not provided or too simple.)

g) Lessons Learned: Describe the concepts and skills you have learned from this programming assignment and any inspiration or motivation you got from this programming assignment. (-10% if not provided or too simple.)

Submission Requirements:

1. Zip all your files including your formal report, all *.java files, and a Java executable file (a jar file, so that I can run your code) into one zipped file with file name: **PA1-Last_Name.zip**, Any file with incorrect file name will be taken off 20%. **[No jar file or your code is not runnable (test it before submission: java -jar filename.jar): -40% if missing]**
2. The deadline to submit/upload your zip file to Canvas is before noon on 4/7/2020. After deadline, the submission link will be closed.
   Warning: DO NOT wait until the last minute to upload your files to Canvas. It has been happening all the time that it took forever to upload files to Canvas during the last hour/minute. No excuse for late submission.

Grading Criteria

Grading is based on the creative design, full functioning, completeness and clarity of codes and the **professional** report. Each grading policy is specifically listed after each functional item above. You should follow the functional requirements to avoid losing points.

Useful Reference Links:

1. Make Java executable Jar file

Note: Eclipse with JDK 8 and under can export project into Runnable Jar, but not JDK 9.