ASSIGNMENT #1:
TRUCK CARGO MANAGEMENT SYSTEM

OBJECTIVES

- Use of simple ADT such as Arrays, Linked Lists, Stacks and Queues.
- Resolve a simple problem.
- Measure algorithm temporal complexity using asymptotic and empirical analysis.

PROBLEM STATEMENT

Develop a Java program named `Tp1` to manage warehouse cargo truck process. A large warehouse has several storage buildings. The program receives as an input the total number of boxes to be transported, the maximum truck capacity, the positions of storage buildings involved in the current cargo and the number of boxes available at each point of cargo. To start, you have to find a service point with the largest number of boxes. The coordinates of this building become the current position of the truck. For each shipment, the program must display the positions of the storage buildings located closest to the truck position, and the number of boxes remaining at the service points. If the distances are the same, you must take the one with the lower latitude. If latitudes are the same for several service points, you have to choose a point with smaller longitude. It is assumed that transportation of all available boxes it is possible from each building to the truck. The number of boxes to be loaded into the truck must not exceed the maximum capacity of the truck. Distances should be calculated using the haversine formula (https://fr.wikipedia.org/wiki/Formule_de_haversine). The Earth radius r is 6371000 metres.

Assumptions and directives. The truck always stays at its specified position, it is rather lifts that load boxes. A lift can deliver all the boxes available at a time. Truck loading is maximized only at the beginning by placing the truck at the service point with the maximum number of boxes. Then, we try to minimize lift distances.

You can use Java library implementations for simple data structures such as Array, List, Stack, and Queue. If you need to use a sorting algorithm for your solution you must implement it. The method sort of the Java library cannot be used.

Example

Suppose we need to transport 15 boxes of merchandises. Truck capacity is measured in terms of number of boxes (15 in the example). The first line of the input file contains this information (15 15). If the number of boxes to transport is bigger than truck capacity, you have to find a solution based on maximum truck capacity. Lines beginning from 2 specify service points of a warehouse with a number of boxes available : there are 4 service points in the current list. Each service point is defined by its position (coordinates, latitude and longitude). Examples of the input and generated result are provided below. To avoid encoding problems and to facilitate automatic correction, all messages have to be written in English with exact syntax (presented below and described in details in the following sections).

```
15 15
2 (45.515399, -73.561996) 5 (45.5092,-73.5682)
8 (45.4383,-73.8205) 4 (45.4977,-73.714)
```

```
Truck position: (45.4383,-73.8205)
Distance:0          Number of boxes:0          Position:(45.4383,-73.8205)
Distance:10611.3    Number of boxes:0          Position:(45.4977,-73.714)
Distance:21193.7    Number of boxes:2          Position:(45.5092,-73.5682)
```

PROGRAM STRUCTURE

The `Tp1` program has to be launched with 2 arguments on the command line: a name of input file and a name of result file.

```
java Tp1 nomfichier1.txt nomfichier2.txt
```

The file `nomfichier1.txt`, an input file, contains total number of boxes to load on the truck and truck capacity in terms of boxes on the first line. All subsequent lines represent available boxes at each service point with the point coordinates.

With `nomfichier1.txt` used as argument (`args[0]`), your program should read from the file `nomfichier1.txt`, do necessary treatment and save the result into the file which name is passed as a second argument, `nomfichier2.txt(args[1])`.

Consider the input file `camion_entrepot.txt`. The file camion`_entrepot.txt` format is as follows.

First line: number of boxes to transport and truck capacity.

Next lines: a set of elements composing of 2 fields:

- Number of boxes at each service point;
- Coordinates, latitude and longitude, of each service point;

All data fields are separating by non-printable characters: tabulation, line escape, space etc.

`camion_entrepot.txt`

```
15 15
2 (45.515399, -73.561996) 5 (45.5092,-73.5682)
8 (45.4383,-73.8205) 4 (45.4977,-73.714)
…
```

Result
Cargo recommendation calculated by your program should be saved in a file with the name specified as a second argument to your program.

```
Truck position: (45.4383,-73.8205)
Distance:0              Number of boxes:0               Position:(45.4383,-73.8205)
Distance:10611.3        Number of boxes:0               Position:(45.4977,-73.714)
Distance:21193.7        Number of boxes:2               Position:(45.5092,-73.5682)
```

Output format: First line: "`Truck position:`", a space (tabulation) followed by the service point coordinates. On each next line: "`Distance:`" followed by the value of calculated distance, a space followed by the string "`Number of boxes:`", the remaining number of boxes at the considered service point and finally, a space, followed by the string "`Position:`" with the corresponding coordinate. All spaces could be replaced by tabulation. You should display the points implicated in the cargo.

Respect this format!!! Automatic correction will be used!!!

Command execution:

```
java Tp1 camion_entrepot.txt res+.txt
```

has to produce the result (`res+.txt`) :

```
Truck position: (45.4383,-73.8205)
Distance:0              Number of boxes:0               Position:(45.4383,-73.8205)
Distance:10611.3        Number of boxes:0               Position:(45.4977,-73.714)
Distance:21193.7        Number of boxes:2               Position:(45.5092,-73.5682)
```

ALGORITHM RUNNING TIME

You have to do two types of temporal complexity analysis of your solution: experimental (slide 6 Algorithm analysis) and theoretical (slides 8-18). The results have to be presented and discussed in a rapport.

SUBMISSION

TP1 electronic submission has to be done by 11:55 p.m. the **12 June 2019 at the latest**. Submit one zipped file containing all source code via StudiUM. A rapport could be submitted electronically or in paper version.

Rapport

A rapport should include following parts:

1. **Auto-evaluation:** Specify if your program works correctly, partially or does not work at all.
2. **Theoretical temporal complexity analysis (worst case, big O notation)**
    - Analysis has to be done on the solution pseudo code or on the developed Java code
3. **Empirical temporal complexity analysis**
    - Graphic demonstrating the algorithm running times for different sizes of the input.

Temporal complexity has to be expressed in terms of the size of the input *n* where *n* indicates the number of service points implicated in cargo process.

GRADING

Grading scheme

Criteria	Description	Ponderation
A.	**Directives**	/ 0.5
B.	**General Appreciation** Program Structure + Code quality	/ 1
C.	**Correctness**	/ 4
D.	**Auto-evaluation (Exactitude)**	/ 0.5
E.	**Running time evaluation :** theoretical and empirical	/ 4
	Total :	10/ 10

Problematic cases: 2 points could be subtracted for the presentation.