ASSIGNMENT #3: Planning of electronic billboards installation

1. Objectives

- Implement an abstract data type Carte based on the graph representation.
- Implement an algorithm of finding a minimum spanning tree

2. PROBLEM STATEMENT

You have to develop a Java programme named `Tp3`. In `Tp3` you have to find a minimum spanning tree for a graph representing a city map.

Mayor's office has decided to install advertising terminals in a neighborhood. It wants to connect all the sites (public places) of the district between them via a wired network. The following data is provided by mayor's office:

- The list of the most visited public places with corresponding streets in the neighborhood;
- The cost of terminal installations on each street segment.

A Street is identified by a set of places. Streets and sites are identified by theirs names:

`rue0 a b 4`.

`rue0` – the street name; **`a`**, **`b`** – the places names; 4 – installation cost.

The neighborhood sites are connected to each other by forming a labeled connected graph where the nodes represent the neighborhood sites, the edges are the segments of the streets that connect the sites to each other and the weights on each segment represent the installation cost. The city wants to connect, in an optimal way, all the sites of the district by an advertising network. The objective of the TP3 is to find a minimum spanning tree (ARMin) of the neighborhood site graph based on Prim-Jarnik algorithm. Edges of the same weight must be processed according to the alphanumeric order of the edge out vertices and in the case of equality of weights and edge "out vertices", use the alphanumeric order of the edge "in vertices".

3. Program structure

Call syntax as well as input and output formats should be preserved.

3.1 Call syntax

The call of your program should be as follows:

```
java Tp3 carte.txt arm.txt
```

The file `carte.txt` specifies a district public places map with the installation cost. The results (a generated minimum spanning tree) have to be saved in the file `arm.txt`

3.2 The file `carte.txt`

The file **`carte`**`.txt` is composed of:

1. Public places list (nodes). Each place is defined by its name (string) on a separate line.
2. Next line - 3 dashes (---) of separation;
3. Following lines: one street segment specified on each line.
   Name of the street (string); a colon (:); two public places names (strings, in and out edge vertices); a number representing the installation cost on the street segment specified on the line and finally a semicolon (;).
4. 3 dashes (---) of the end.

Example (**`carte0.txt`**) :

```
a
b
c
d
e
f
g
h
i
---
rue0 : a b 4 ;
rue1 : a h 8 ;
rue2 : b h 11 ;
rue3 : b c 8 ;
rue4 : c i 2 ;
rue5 : c f 4 ;
rue6 : c d 7 ;
rue7 : d f 14 ;
rue8 : d e 9 ;
rue9 : e f 10 ;
rue10 : f g 2 ;
rue11 : g i 6 ;
rue12 : g h 1 ;
rue13 : h i 7 ;
---
```

3.3 Tp3 output format

The found spanning tree has to be saved in the arm.txt file in the following format.

A sites list, one site on each line. Then, on each line, you have to save street segments with corresponding information. You have to follow the alphanumeric order of segment predecessor names and in case of equality you have to compare the successors names. On each line you save a segment (edge):

a) Street name : `rue0`
b) Sites names : `a b`
c) A label (cost) : `4`

After displaying all segments:

- 3 dashes (---) of separation;
- The spanning tree cost on the last line.

Output example:

```
a
b
c
d
e
f
g
h
i
rue0 a b 4
rue1 a h 8
rue6 c d 7
rue4 c i 2
rue8 d e 9
rue5 f c 4
rue10 g f 2
rue12 h g 1
---
37
```

5. Tests

The correction will be made automatically, so the respect of call and output syntaxes is mandatory.

6. Submission

The TP3 should be submitted 31 July at 23h55 at the latest.

There will be a penalty of 10% per day for a late submission.

6.1 Report

- Time complexity analysis in Big O notation. Don’t forget the justification.
- The report must be submitted electronically on StudiUM.

6.2 Source code submission

StudiUM

7. Grading

- 10% of the final grade.

Grading scheme

Criteria	Description	Score
A.	**Directives**	/ 1
B.	**General Appreciation** Program Structure + Code quality	/ 2
C.	**Correctness**	/ 4
D.	Efficiency	/ 0.5
E.	**Algorithm analysis** Temporal complexity with justification	/ 2.5
	Total :	10/ 10