Projet : métro

Un programme qui vous donne l'itinéraire le plus rapide de votre trajet dans Paris.

Nom du programme : Itinéraire

Projet en Scheme

Par : Thanusan et Aljoscha

1)Le Programme Itinéraire

a)Introduction à notre programme

Notre programme se nomme itinéraire. Ce programme vous donne l'itinéraire le plus rapide entre une station de départ et une station d'arrivée qui font parti des lignes que nous avons choisi d'intégrer dans notre programme (la ligne1, la ligne 2, la ligne 6, la ligne 4 et enfin la ligne 5).

B)L'exécution de notre programme.

L'exécution de notre programme est simple :

1- Ouvrir le fichier nommé " itinéraire.scm " dans DrRacket
2- Cliquer sur " Exécuter "
3- Dans la fenêtre d'interaction, veuillez écrire :

(itinéraire le-nom-de-votre-station-de-départ le-nom-de-votre-station-d´arrivée)

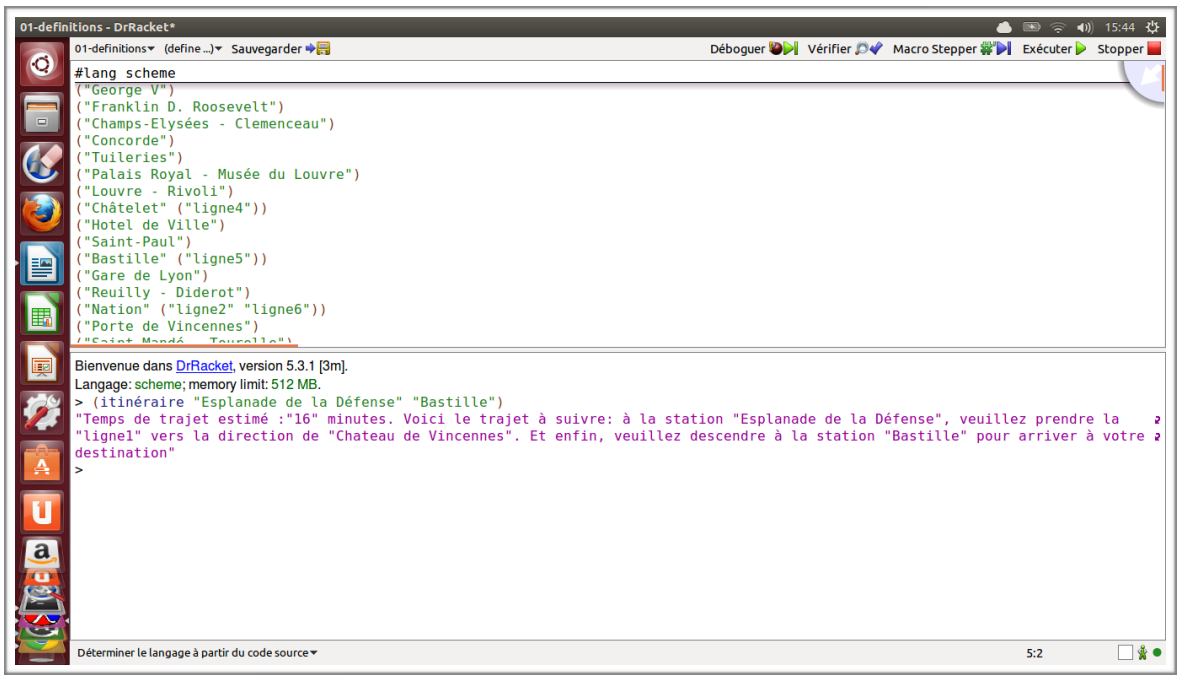
ATTENTION : le nom de votre station doit être placer entre des guillemets doubles ("station") car sinon le programme ne comprendra pas les stations que vous avez entré.

Exemple: pour Châtelet, vous devrez écrire "Châtelet"

pour Nation, vous devrez écrire "Nation"

C) Quels que exemples d'exécution

1)un trajet sur une ligne



Ce qui est affiché :

Temps du trajet : 16 minutes

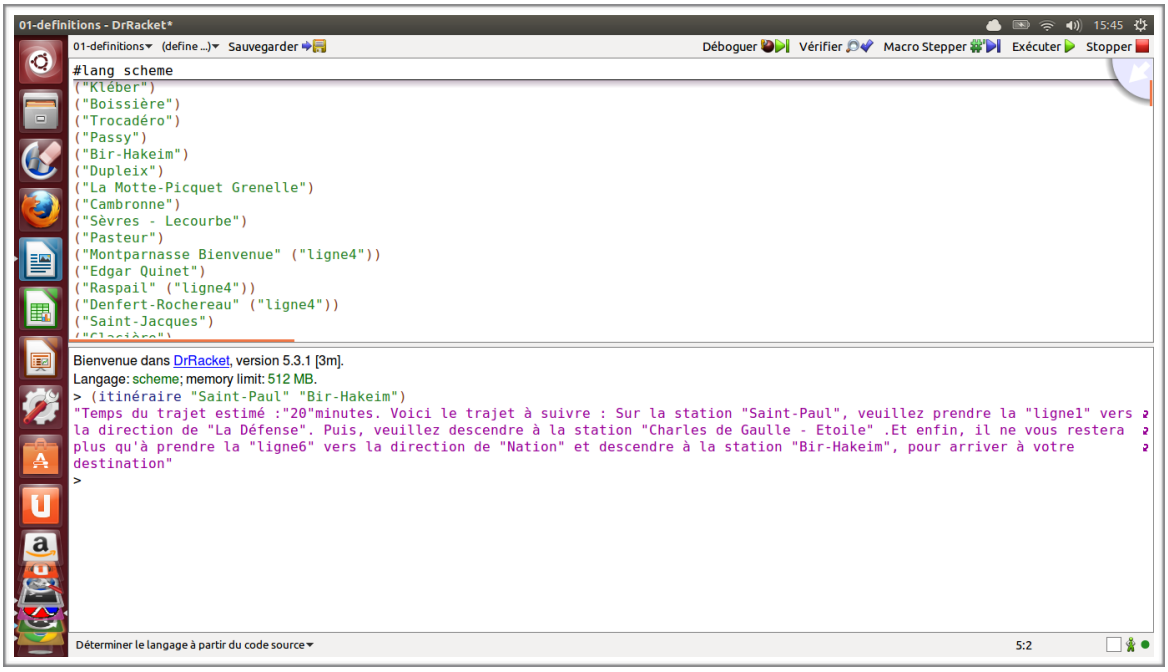
Station de départ : Esplanade de la Défense

La ligne qu'il faut prendre : ligne 1

La direction qu'il faut prendre sur la ligne : Château de Vincennes

Station d'arrivée : Bastille

2) un trajet sur deux lignes



Ce qui est affiché :

Temps du trajet : 20 minutes

Station de départ : Saint-Paul

Ligne de départ : ligne 1

Direction de la ligne de départ : La Défense

Station qui sert de correspondance pour aller d'une ligne de départ à une ligne d'arrivée : Charles de Gaule - Étoile

Ligne d'arrivée: ligne 6

Direction de la ligne d'arrivée : Nation

Station d'arrivée: Bir Hakeim

2)information sur l'idée et la structure de notre programme

A)Information sur l'idée de notre programme

Nous avons passé environ deux semaines pour trouver une idée idéale pour notre projet. Toutes les idées que nous avons utilisé ont montré leur limites. Nous avons donc commencé à analyser chaque trajet sur les différentes lignes.
Et nous avons remarqué plusieurs situations difficiles.

Par exemple : Sur un trajet dans une seule ligne, nous avons remarqué qu'il y'avait des stations de départ et des stations d'arrivée qui se trouvaient sur plusieurs lignes en mêmes temps (comme le cas de Nation et Charles de Gaule - Étoile : nous pouvons trouvé ces deux stations sur la ligne 1, 2 et 6)

Puis, nous avons remarqué sur les trajets en deux lignes, qu'il y'avait des stations de départ et d'arrivée qui se trouvaient sur plusieurs lignes, et qu'il faut prendre en compte toutes ces lignes pour donner le trajet le plus court à l'utilisateur. D'ailleurs, sur les trajets qui nécessitent deux lignes, nous avons vu qu'il y'avait parfois plusieurs

stations de correspondances qui nous permet d'accéder à la ligne d'arrivé, donc le programme doit prendre en compte toutes les stations de correspondance.

Donc, nous avons décidé de créer un programme qui cherche touts les trajets possible, et qui s'adapte à toute les situations et à toutes les modifications (par exemple si on ajoute une station, nous n'avons pas besoin de changer tout le code source).

Nous avons aussi rencontré des problèmes lors de la mise en œuvre de notre idée.

En effet, nous avons voulu créer 3 programmes sur les trajets, un programme pour un trajet sur une ligne, un programme pour un trajet sur deux lignes et un programme pour un trajet sur 3 lignes.

Malgré, la création d'une dizaine de fonction pour le programme pour un trajet sur 3 lignes, nous l'avons retiré car il montrait plusieurs limites et car nous avons remarqué qu'il n'était pas nécessaire pour les lignes que nous utilisons.

B)La structure de notre projet

Après avoir définis les différentes lignes que nous allons utilisé, nous avons définis toutes des petites fonctions (comme lines, truelinefinder, existe?, listcounter-for-travel, only-station).

Nous avons créé ces petites fonctions afin de créer des fonctions (tel que correspondances, intervalle, lignesearcher, quel-direction, Ou-est-l-dans-ml) qui peuvent servir d'outils pour les fonctions sur les trajets.

Puis, nous avons crée la fonction pour un trajet sur une ligne et la fonction pour un trajet sur deux lignes après avoir crée de nombreux sous-fonctions (que nous expliquerons dans la partie sur la liste des principales fonctions).

Et enfin, nous avons assemblé toutes les fonctions en un programme qui se nomme itinéraire.

3)Les informations sur les fonctions

A)les fonctions les plus délicates

Les fonctions les plus délicates sont :

La définition des lignes

La fonction intervalle (et ses sous fonctions)

La fonction lignesearcher

La fonction Ou-est-l-dans-ml (et ses sous fonctions)

La fonction quel-direction (et ses sous fonctions)

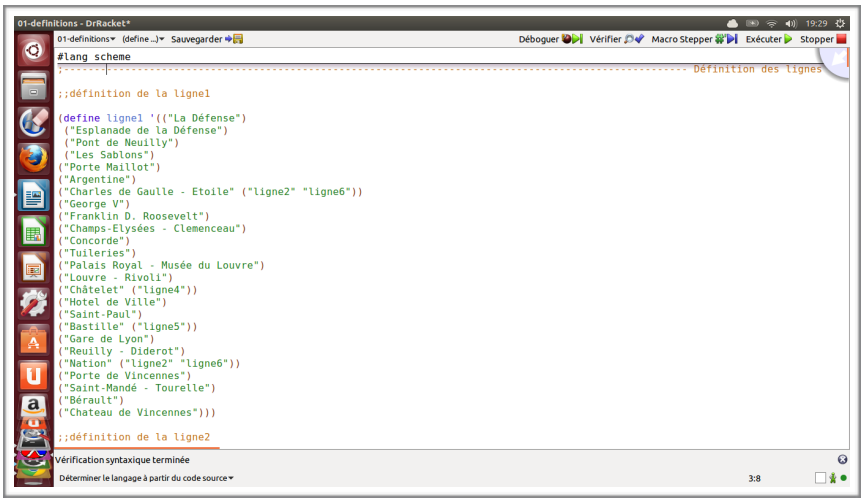
La fonction oneline-info (et ses sous fonctions)

La fonction finaltwolines (et ses sous fonctions)

Comme les fonctions les plus délicates à coder font partie des principales fonctions, nous expliquerons leur fonctionnement sur la prochaine partie.

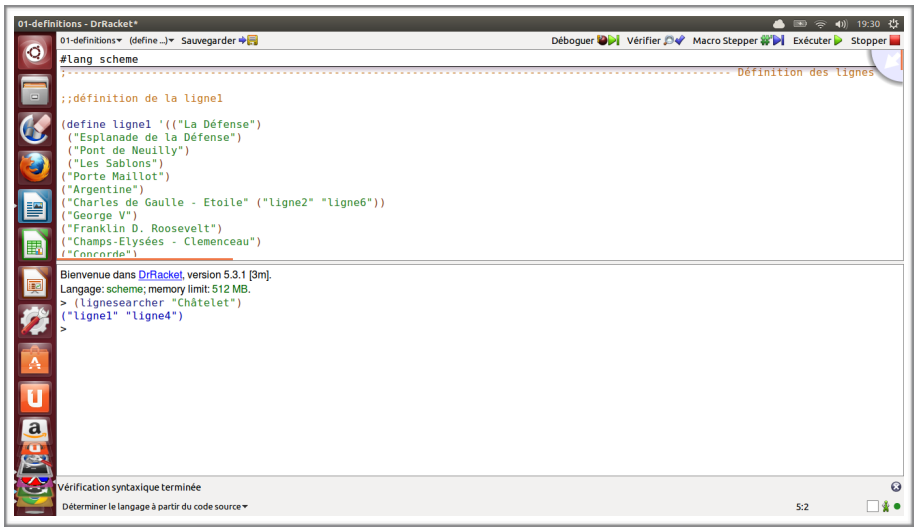
B) La liste des principales fonctions

1- La définitions des lignes



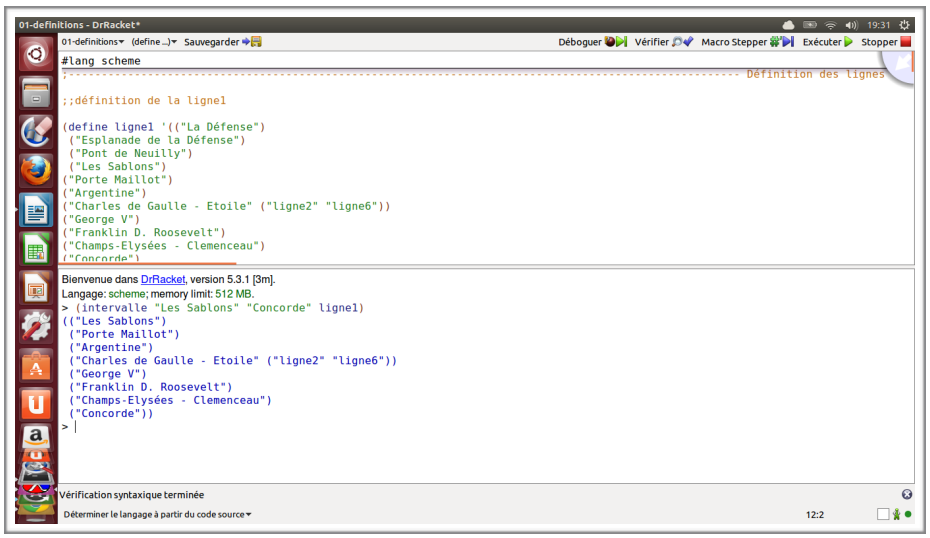
Nous avons eu l'idée de définir les lignes comme sur l'image ci dessus, après avoir défini plusieurs fois et de plusieurs manières différentes les lignes. Cette façon de définir les lignes nous a permis d'avoir moins de limite.

2-lignesearcher



lignesearcher nous renvoie la liste de toutes les lignes où l'on peut trouver la station entrée.

3-intervalle



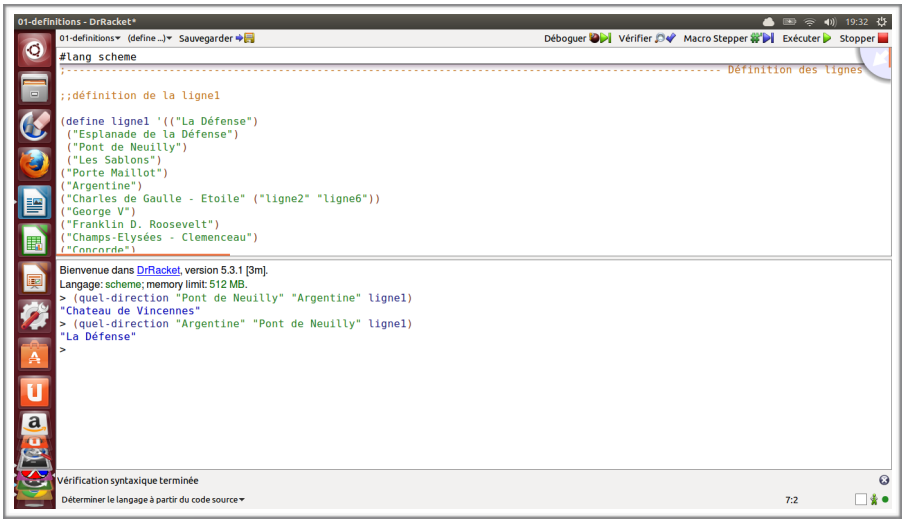
La fonction intervalle nous donne une liste des stations qui sont entre la station de départ (d) et la station d'arrivée (a) dans une ligne entrée (l). Cette fonction nous donne la liste des stations dans l'ordre de départ et d'arrivée (c'est à dire en respectant la direction). Plusieurs sous fonctions ont été nécessaires pour l'aboutissement du fonction intervalle:

-enleve-avant-s: cette fonction nous donne la liste des stations qui sont avant la station entrée (d) dans la ligne entrée (l). La station entrée n'apparaitra pas dans la liste des stations que cette fonction nous renverra.

-enleve-après-s: Cette fonction nous donne la liste des stations qui sont après la station entrée (a) dans une ligne entrée (l). Et la station entrée n'apparaitra pas dans la liste des stations que cette fonction renverra.

-enleve-avant-s-et-après-s : Cette fonction nous donne la liste des stations qui ne sont pas dans l'intervalle de la station entrée dans (d) et de la station entrée dans (a) dans une ligne entrée (l). D'ailleurs, cette fonction nous donne une liste dans l'ordre de départ et d'arrivée.

4- quel-direction

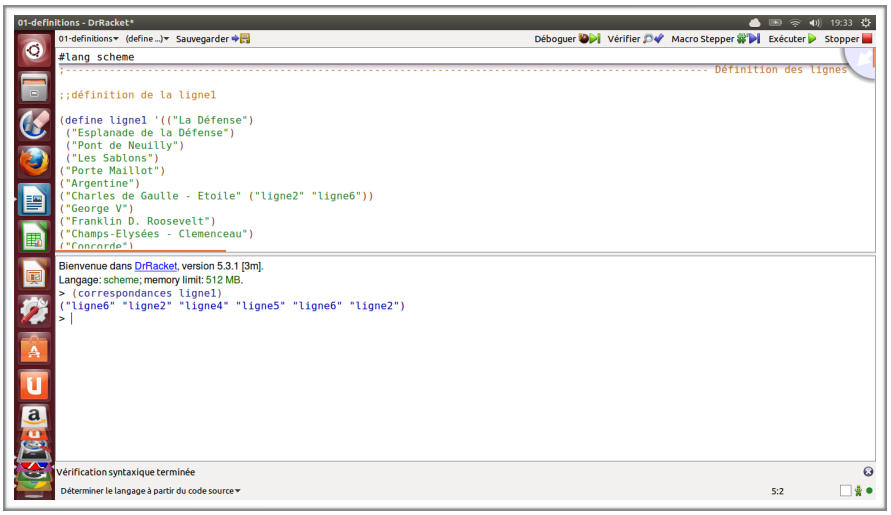


Ce programme nous donne la direction à prendre selon le trajet à faire sur une ligne. Plusieurs sous fonctions ont été nécessaire pour l'aboutissement de ce programme :

-bonnedirection? : Ce prédicat nous renvoie true si la station entrée dans (d) est avant la station entrée dans (a) (de gauche à droite) et false si la station entrée dans (a) est avant la station entrée dans (d) (de gauche à droite) dans une ligne entrée (l). En effet, cette fonction donne des détails sur les directions.

-compteurbd : ce compteur n'est pas vraiment un compteur ça il nous donne l'emplacement de la station entrée en (d) par rapport à la première station (de gauche à droite) dans la ligne entrée dans (l)

5-correspondances



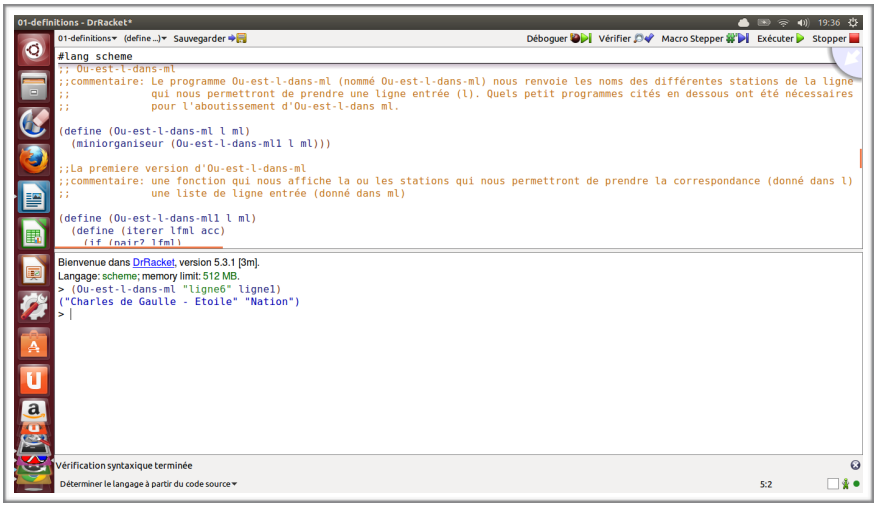
Le programme correspondances affiche la liste des lignes que l'on peut trouver en correspondances dans une ligne entrée. Plusieurs pré-versions cités en dessous ont étés nécessaire pour l'aboutissement du programme correspondances :

-correspondances version1 : La première version du "correspondances" affiche les stations dans lesquelles il y'à au moins une correspondance.

-correspondances version2 : La deuxième version du "correspondances" affiche la première version de correspondance sans les noms des stations.

correspondances version3 :La troixième version du "correspondances" affiche ce qu'affiche correspondances2 en mieux (c'est à dire avec moins de parenthèses).

6-Ou-est-l-dans-ml

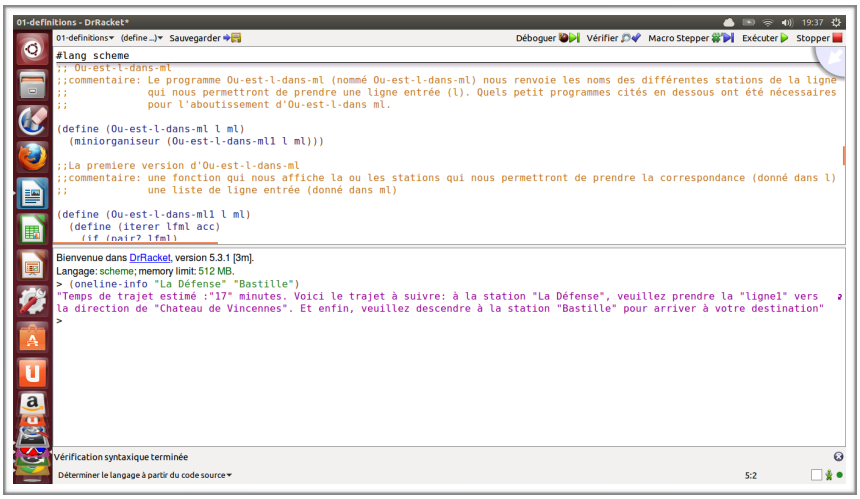


Le programme Ou-est-l-dans-ml (nommé Ou-est-l-dans-ml) nous renvoie les noms des différentes stations de la ligne entrée dans (ml), qui peuvent servir de correspondance pour accéder à une ligne entrée dans (l). Quels petit programmes cités en dessous ont été nécessaires pour l'aboutissement d'Ou-est-l-dans ml :

-Ou-est-l-dans-ml1 : une fonction qui nous affiche la ou les stations qui nous permettront de prendre la correspondance (donné dans l) dans une liste de ligne entrée (dans ml).

-miniorganiseur : est une sous fonction pour Ou-est-l-dans-ml. Ce programme nous permet d'afficher seulement les noms des stations dans la liste que donne Ou-est-l-dans-ml1 (il faut rentrer cette liste dans l)

7) oneline-info



Son rôle est de trier les différentes informations de la liste que minimum-oneline-organisateur lui renvoie, pour pouvoir afficher à l'utilisateur le trajet à faire et le temps de trajet (pour un trajet sur une seule ligne). Ce n'était pas difficile de coder cette fonction, mais c'était difficile de coder ses sous fonctions qui ont été indispenseable:

-oneline? : Ce prédicat renvoie true si on peut trouver au moins une fois la station entrée dans départ (d) et la station entrée dans arrivée (a) sur une même ligne. Sinon, ce prédicat nous renverra false.

-onelinev1: Ce programme nous donne la liste des lignes où l'on peut trouver les deux stations entrées (dans (d) et (a)).

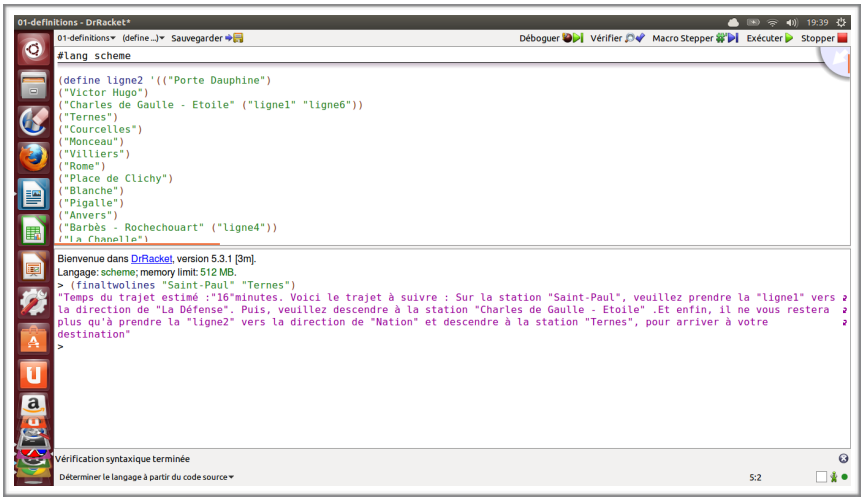
-intervalle-oneline : ce programme affiche la liste des stations qui sont entre la station entrée dans le départ (d) et la station entrée dans l'arrivée (a) pour chaque ligne de la liste que onelinev1 renverra pour les deux stations (d et a) entrées. (Les deux stations d et a, seront dans la ou les listes).

-compteur-intervalle-oneline : Ce programme affiche dans l'ordre, le nombre de stations que contient chaque sous-listes de la liste intervalle-oneline. Ce programme peut servir d'horaire pour les trajets correspondances.

-oneline-organisateur : Ce programme est un organisateur. En effet, il trie les informations donnée par le programme onelinev1, compteur-intervalle-oneline, et intervalle-oneline dans l'ordre qu'il le faut. le tri que ce programme nous donnera, nous permettra d'avoir le temps du trajet, la ligne d'un trajet, ainsi que l'intervalle entre le départ et l'arrivée sur cette ligne.

-minimum-oneline-organisateur: Ce programme affiche la sous-liste qui possède le minimum de temps du trajet parmi les différentes sous-listes que contient la liste donné par le programme oneline-organisateur.

8-finaltwolines



Ce programme trie les différentes informations donné par le programme minimum-d'-info-trajet-pour-twolines afin de donner à l'utilisateur les informations nécessaires sur le trajet et le temps du trajet. Ce n'était pas difficile de coder ce programme mais c'était difficile de coder ses sous fonctions qui ont été indispensable pour l'aboutissement de ce programme:

-twolines? : Ce prédicat a pour rôle d'envoyer true si on peut utiliser la fonction quel-stations-pour-twolines pour un trajet. Si ce trajet n'est pas possible par la fonction quel-stations-pour-twolines, il renverra false.

- twolines-v1: est un programme qui cherche les différentes lignes dans les quelles les stations (d) et (a) peuvent se trouver et il regarde si la ligne d'arrivée est dans les correspondances de la ligne de départ. Si oui, il va créer une sous-liste avec la ligne de départ et d'arrivée pour faciliter la création des prochaines sous fonctions. Ce programme est très intéressant. En effet, il est une solution pour plusieurs problèmes que nous avons rencontré pour la création d'un programme qui gère un trajet sur deux lignes .(quelques exemples des problèmes : la station de départ ou d'arrivé peut être sur plusieurs lignes. Parfois, sur une ligne de départ, nous pouvons trouver plusieurs correspondances pour accéder à la ligne d'arrivée...)

-quel-stations-pour-twolines : Ce programme cherche les stations qui permettent d'accéder d'une ligne de départ à la ligne d'arrivée pour chaque sous liste de la liste que twolines-v1 donne. D'ailleurs, ce programme nous donne une liste propre avec la liste des stations appropriés à chaque sous liste de twolines-v1.

-quel-stations-pour-twolines-v2 :Ce programme est une évolution du programme quel-stations-pour-twolines. En effet, il a pour rôle de trier les informations de quel-stations-pour-twolines, afin de faciliter la version finale de twolines.

Exemple: si la premiere version du quel-stations-pour-twolines affiche :

'(("g" "t") ("ligne2" "ligne5"))

la deuxieme affichera : '(("g" ("ligne2" "ligne5")) ("t" ("ligne2" "ligne5"))).

-intervalle-twolines :Ce programme est très intéressant. En effet, ce programme est la version adapté du programme intervalle pour le programme finaltwolines, on peut le servir pour avoir la liste des stations qui sont entre la station de départ et la station de correspondance dans la ligne de départ, ou la liste des stations qui sont entre la station

de correspondance et la station d'arrivée dans la ligne d'arrivée. C'est le programme indispensable pour la version finale de l'idée twolines (programme sur un trajet en deux lignes).

-intervalle-de-d-à-c-puis-de-c-à-a: Ce programme introduit l'une des meilleures évolution qui permet d'accéder à la version finale finaltwolines. En effet, grace à l'aide du programme intervalle-twolines, ce programme donne pour chaque sous liste du programme quel-stations-pour-twolines-v2, l'intervalle entre la station de départ et la station de la correspondance dans la ligne de départ, et l'intervalle entre la station de la correspondance et la station d'arrivée dans la ligne d'arrivée sous forme de sous liste en respectant l'ordre des sous listes.

-compteur-twolines : Ce programme est une petite fonction définie spécialement pour calculer le temps du trajet d'une sous-liste de la liste donné par le programme intervalle-de-d-à-c-puis-de-c-à-a.

-horaire-twolines : Ce programme nous donne en effet la liste du programme compteur-twolines pour chaque sous liste de la liste donné par le programme intervalle-de-d-à-c-puis-de-c-à-a. (Il a la meme fonction que map avec compteur-twolines sur une liste de sous-liste donné par le programme intervalle-de-d-à-c-puis-de-c-à-a).

-horaire-avec-chaque-trajet-twolines : Ce programme nous donne une liste avec l'horaire respectif des sous-listes de la liste donné par le programme intervalle-de-d-à-c-puis-de-c-à-a grâce au programme horaire-twolines.

-info-trajet-pour-twolines : Ce programme est indispensable pour avoir des informations sur un trajet avec deux lignes. En effet, ce programme crée à l'aide du programme quel-stations-pour-twolines-v2, une sous-liste qui contient dans elle, deux autres sous-listes, dont la premiere qui contient dans l'ordre, le nom de la station de correspondance, la ligne d'arrivée et la ligne de départ et la deuxieme sous-liste qui est la sous-liste du programme horaire-avec-chaque-trajet-twolines qui correspond à la première sous-liste. Ce programme aura autant de sous-liste que le programme horaire-avec-chaque-trajet-twolines.

-minimum-d'-info-trajet-pour-twolines: Ce programme donne la sous liste qui contient le trajet le plus court parmi les sous-listes donné par le programme info-trajet-pour-twolines.