Travail pratique #2 : Liste d'épicerie

Objectifs

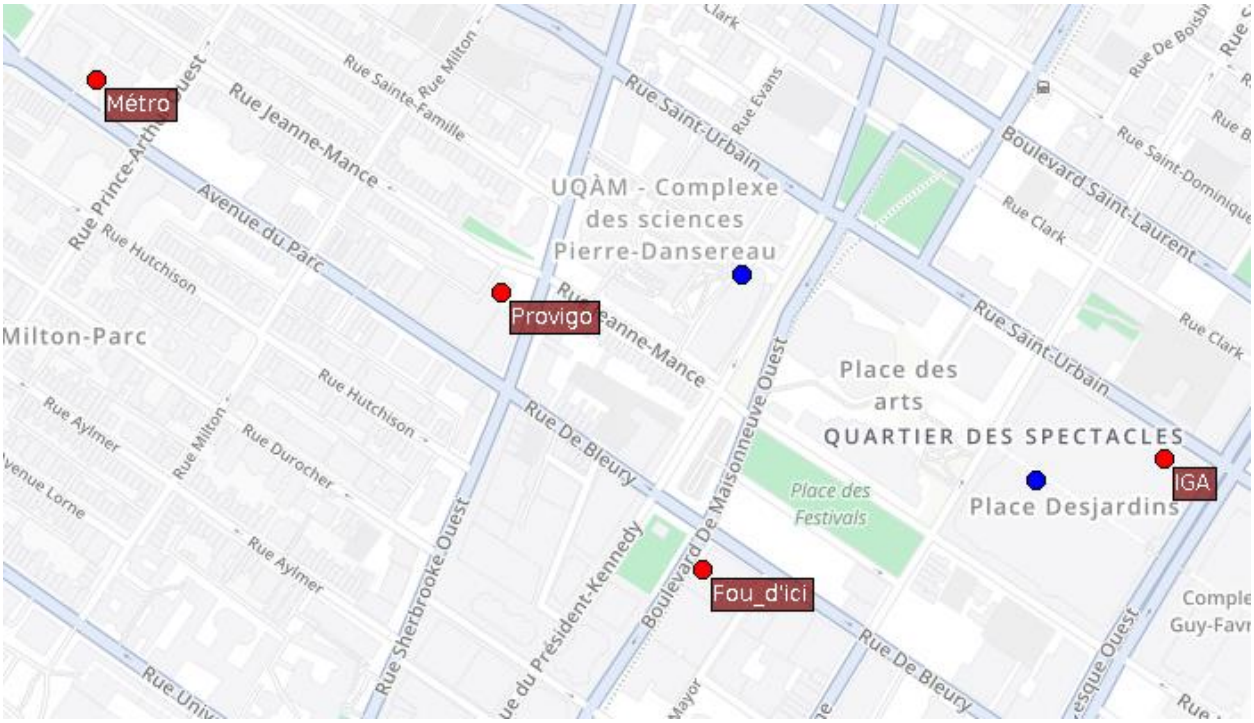
- Implémenter un type générique d'arbres binaires de recherche de type AVL.
- Appliquer les arbres binaires de recherche à une problématique.
- Écrire un programme efficace en temps et en mémoire.

Problématique

Vous devez écrire un programme C++ nommé `tp2` qui recommande à des clients le trajet le plus court pour ramasser les produits sur leur liste d'épicerie. Une liste d'épicerie est composée de noms de produit avec leur quantité requise. Chaque client part de sa résidence, visite une ou plusieurs épiceries, et retourne ensuite à sa résidence. Chaque épicerie a un nom, est située à un emplacement et stocke un inventaire. Chaque produit dans l'inventaire est une date d'expiration.

Exemple

Soit la carte ci-bas où il y a 4 épiceries à proximité du Complexe des sciences de l'UQAM.



Supposons l'inventaire des 4 épiceries comme suit.

Fou d'ici	**IGA**	**Métro**	**Provigo**
• Fraises	• Chocolat	• Chocolat	• Bleuets
• Lait	• Lait	• Lait	• Lait
• Pain	• Pommes	• Oranges	• Pain

Supposons les requêtes suivantes.

1. Alice est au PK (premier point blue) et veut du Lait. Le trajet le plus court est d'aller au Provigo qui est légèrement plus près que le Fou d'ici.
2. Benoit est au PK et veut du Lait et des Pommes. Son trajet le plus court est d'aller au IGA pour récupérer ces deux produits.
3. Cloé est au PK et veut du Chocolat. Son trajet le plus court est d'aller au IGA.
4. Daniel est au PK et veut du Chocolat et du Pain. Son trajet le plus court est d'aller au IGA et au Fou d'ici, ce qui est légèrement plus court que de passer au Métro et au Provigo.
5. Eugénie est au PK et veut du Chocolat, du Pain et des Oranges. Son trajet le plus court est d'aller au Métro.

6. François est au PK et veut du Lait et tous les fruits (Bleuets, des Fraise, des Oranges et des Pommes). Son trajet le plus court est d'aller dans l'ordre (ou désordre) : IGA, Fou d'ici, Provigo et Métro.

Hypothèses

1. Comme pour le TP1, les déplacements se font en ligne droite sur la surface la Terre. La distance entre deux points (coordonnées latitude et longitude) sur la carte se calcule avec la fonction `PointST::distance()` fournie. Celle-ci retourne la longueur en mètres en faisant l'hypothèse que la Terre est une sphère parfaite d'un rayon de 6371 km.
2. Lorsqu'un client visite plus d'une épicerie, il achète les produits sur sa liste dès la première occasion, et ce, selon l'ordre de visite des épiceries calculé. Par exemple, dans l'exemple précédent, François achètera le Lait à la première épicerie sur son trajet, soit au IGA.
3. Les produits expirent à la date indiquée. Les produits expirés sont retirés de l'inventaire dès leur expiration.
4. Lorsqu'un client achète un produit, il prend toujours les exemplaires ayant la date d'expiration la plus tardive.

Structure du programme

Un squelette de départ recommandé, mais optionnel, est disponible dans tp2.zip.

Syntaxe d'appel du programme tp2

Votre programme doit pouvoir être lancé en ligne de commande avec la syntaxe suivante :

```
./tp2 [nomfichier]
```

où `nomfichier` est optionnel.

Si `nomfichier` est spécifié, alors votre programme doit lire dans `nomfichier` au moyen d'un flux de lecture C++ `std::ifstream`. Sinon, votre programme doit lire dans l'entrée standard (*stdin*) au moyen du flux d'entrée C++ `std::cin`. À noter que le squelette implémente déjà cela.

Les résultats produits par votre programme doivent être écrits dans la sortie standard (*stdout*) à l'aide du flux de sortie C++ `std::cout`.

Format d'entrée et de sortie

Le flux d'entrée est une suite de commandes. Chaque commande débute avec un mot-clé en majuscules spécifiant son type.

Commande	Syntaxe et description

DATE	`"DATE" date ";"`
	
	Spécifie la nouvelle date courante. Il est garanti que la nouvelle date est plus grande (>) que la précédente. La première date ne sera jamais avant le 2000-01-01.
	
	La date courante est importante pour déterminer les produits expirés.
	
	Toutes les dates dans le TP2 sont spécifiées dans le format `AAAA-MM-JJ`. Un début d'implémentation d'une classe `Date` est fourni dans les fichiers `date.h` et `date.cpp`. Vous pouvez la compléter ou créer votre propre type `Date`. Il n'est pas nécessaire de valider les dates lues. Ainsi, vous n'avez pas à gérer le nombre de jours par mois et les années bissextiles.
	
	Exemple d'entrée:
	
	`DATE 2017-10-17 ;`
	
	Exemple de sortie:
	
	`OK`

PLACER	`"PLACER" nom_épicerie position`
	
	Spécifie l'emplacement géographique d'une épicerie à une position décrite par des coordonnées (latitude,longitudes) en degrés. Le nom d'épicerie est unique.
	
	Exemple d'entrée:
	
	`PLACER Foudici (45.506873,-73.568921) ;`
	`PLACER IGA (45.507798,-73.563369) ;`
	`PLACER Métro (45.510993,-73.576233) ;`
	`PLACER Provigo (45.509204,-73.571362) ;`
	
	Exemple de sortie:
	
	`OK`
	`OK`
	`OK`
	`OK`

<table>
<tr><td>APPROV</td><td>

```
"APPROV" nom_épicerie ":" (nom_produit quantité expiration)*
";"
```

Spécifie un réapprovisionnement de produits à une épicerie. Il n'y aura pas d'approvisionnement d'une épicerie non placée.

Exemple d'entrée:

```
APPROV Foudici :
Fraises 5       2017-10-28
Lait    5       2017-10-28
Pain    5       2017-10-28
;
APPROV IGA :
Chocolat        3       2017-10-30
Lait    4       2017-10-30
Pommes  4       2017-10-30
;
APPROV Métro :
Chocolat        2       2017-10-31
Lait    2       2017-10-31
Oranges 3       2017-10-31
;
APPROV Provigo :
Bleuets 4       2017-11-01
Lait    4       2017-11-01
Pain    3       2017-11-01
;
```

Exemple de sortie:

```
OK
OK
OK
OK
```

Un produit peut apparaît plusieurs fois, et ce, avec des dates d'expiration pouvant être différentes. Exemple:

```
APPROV Foudici :
Fraises 3       2017-10-27
Fraises 2       2017-10-28
Fraises 2       2017-10-29
Fraises 1       2017-10-30
Fraises 1       2017-10-30
;
```

</td></tr>
<tr><td>RECOMMANDER</td><td>

```
"RECOMMANDER" position nbMaxÉpicerie
   distanceTotalMax ":" (nom_produit quantité)* ";"
```

Calcule et recommande le trajet le plus court pour une liste d'épicerie d'un client. Le client est à une position spécifiée en format (latitude,longitude). Les produits **ne** sont **pas** retirés de l'inventaire selon les quantités

</td></tr>
</table>

demandées. Si le trajet contient plus d'une épicerie, les produits sont achetés dès la première occasion. En sortie, la commande affiche la distance totale parcourue, arrondie au mètre près, suivi de la lettre `m`, un espace, et enfin la liste des épiceries dans l'ordre de visite.

Dès qu'un produit sur la liste n'est pas en quantité suffisante, la liste est réputée impossible. Dans ce cas, il faut afficher ~~`0m IMPOSSIBLE`~~ `IMPOSSIBLE` en sortie.

Exemple d'entrée:

```
RECOMMANDER (45.509339,-73.568465) 4 999000 :
 Lait 1 ;
RECOMMANDER (45.509339,-73.568465) 4 999000 :
 Lait 1 Pommes 1 ;
RECOMMANDER (45.509339,-73.568465) 4 999000 :
 Chocolat 1 ;
RECOMMANDER (45.509339,-73.568465) 4 999000 :
 Chocolat 1 Pain 1 ;
RECOMMANDER (45.509339,-73.568465) 4 999000 :
 Chocolat 1 Pain 1 Oranges 1 ;
RECOMMANDER (45.509339,-73.568465) 4 999000 :
 Bleuets 1 Fraises 1 Oranges 1 Pommes 1 ;
RECOMMANDER (45.509339,-73.568465) 4 999000 :
 Oranges 1 ;
RECOMMANDER (45.507610,-73.564925) 4 999000 :
 Fraises 1 ;
RECOMMANDER (45.507610,-73.564925) 4 999000 :
 Fraises 100 ;
```

Exemple de sortie:

```
452m  Provigo
865m  IGA
865m  IGA
1154m  Foudici IGA
1287m  Métro Provigo
2260m  IGA Foudici Provigo Métro
1265m  Métro
644m  Foudici
IMPOSSIBLE
```

S'il existe plusieurs solutions optimales, il suffit d'afficher l'une d'elles. C'est notamment le cas dès que `nbMaxÉpicerie` > 1, où l'ordre inversé d'une solution est aussi une autre solution valide. Par exemple, «`1154m IGA Foudici`» est une solution optimale et tout aussi valide que «`1154m Foudici IGA`» dans l'exemple ci-haut. Avec `nbMaxÉpicerie` = 1, il pourrait avoir plus d'une solution optimale dans de très rares cas où il y aurait plus d'une épicerie satisfaisante à distance égale du client.

RAMASSER	`"RAMASSER" (nom_produit quantité)* ";" nom_épicerie* ";"`
	
	Ramasse les produits demandés en visitant les épiceries dans l'ordre spécifié. Les produits doivent être ramassés dès que possible (voir hypothèse 2 à la section 2.2).
	
	En sortie, la commande affiche "COMPLET" si et si seulement si tous les produits ont pu être récupérés. Sinon, "MANQUE" est affiché suivi de la liste des produits manquants avec leur quantité manquante.
	
	Exemple d'entrée:
	
	`RAMASSER Chocolat 2 Lait 3 Pommes 3 ; IGA ;`
	`RAMASSER Chocolat 1 Lait 1 Pommes 2 ; IGA ;`
	
	Exemple de sortie:
	
	`COMPLET`
	`MANQUE Pommes 1 ;`

INVENTAIRE	`"INVENTAIRE" nom_épicerie ";"`
	
	Affiche sur une ligne l'inventaire courant dans l'épicerie demandée. La ligne est terminée par un point-virgule. Les produits en rupture de stock (quantité=0) ne devraient pas être affichés.
	
	Exemple d'entrée:
	
	`INVENTAIRE Foudici ;`
	`INVENTAIRE IGA ;`
	
	Exemple de sortie:
	
	`Fraises 5 Lait 5 Pain 5 ;`
	`;`

Les commandes doivent être traitées dans leur ordre d'arrivée (ordre d'apparition dans le fichier d'entrée). Une fois une commande lue, il faut la traiter immédiatement afin d'afficher son résultat avant de pouvoir lire la prochaine commande.

Pour faciliter le *parsing* au moyen de `std::cin>>`, il y a au moins un espace blanc (espace, tabulation ou retour de ligne) après chaque chaine de caractère ou nombre. Les noms d'épicerie et de produit ne contiennent jamais d'espace blanc.

Exemples d'utilisation

Exemple 1

```
./tp2 < exemple1.txt > exemple1+.txt
```

exemple1.txt

```
DATE 2017-10-17 ;
PLACER Foudici (45.506873,-73.568921) ;
PLACER IGA (45.507798,-73.563369) ;
PLACER Métro (45.510993,-73.576233) ;
PLACER Provigo (45.509204,-73.571362) ;
APPROV Foudici : Fraises 5 2017-10-28
    Lait 5 2017-10-28 Pain 5 2017-10-28 ;
APPROV IGA : Chocolat 3 2017-10-30
    Lait 4 2017-10-30 Pommes 4 2017-10-30 ;
APPROV Métro : Chocolat 2 2017-10-31
    Lait 2 2017-10-31 Oranges 3 2017-10-31 ;
APPROV Provigo : Bleuets 4 2017-11-01
    Lait 4 2017-11-01 Pain 3 2017-11-01 ;
RECOMMANDER (45.509339,-73.568465) 4 999000
:
 Lait 1 ;
RECOMMANDER (45.509339,-73.568465) 4 999000
:
 Lait 1 Pommes 1 ;
RECOMMANDER (45.509339,-73.568465) 4 999000
:
 Chocolat 1 ;
RECOMMANDER (45.509339,-73.568465) 4 999000
:
 Chocolat 1 Pain 1 ;
RECOMMANDER (45.509339,-73.568465) 4 999000
:
 Chocolat 1 Pain 1 Oranges 1 ;
RECOMMANDER (45.509339,-73.568465) 4 999000
:
 Bleuets 1 Fraises 1 Oranges 1 Pommes 1 ;
RECOMMANDER (45.509339,-73.568465) 4 999000
:
 Oranges 1 ;
RECOMMANDER (45.507610,-73.564925) 4 999000
:
 Fraises 1 ;
RECOMMANDER (45.507610,-73.564925) 4 999000
:
 Fraises 100 ;
RAMASSER Chocolat 2 Lait 3 Pommes 3 ; IGA ;
RAMASSER Chocolat 1 Lait 1 Pommes 2 ; IGA ;
INVENTAIRE Foudici ;
INVENTAIRE IGA ;
DATE 2017-11-02 ;
INVENTAIRE Foudici ;
INVENTAIRE IGA ;
INVENTAIRE Métro ;
```

exemple1+.txt

```
OK
OK
OK
OK
OK

OK

OK

OK

452m  Provigo

865m  IGA

865m  IGA

1154m  Foudici IGA

1287m  Métro Provigo

2260m  IGA Foudici Provigo
Métro

1265m  Métro

644m  Foudici

IMPOSSIBLE
COMPLET
MANQUE Pommes 1 ;
Fraises 5      Lait 5  Pain 5  ;
;
OK
;
;
;
;
```

`INVENTAIRE Provigo ;`	

Autres exemples

Il y a d'autres exemples à la fin de l'énoncé.

Contraintes

Librairies permises

Vous devez implémenter et utiliser vos propres structures de données. Pour l'instant, l'utilisation des conteneurs de la librairie standard de C++ (*Standard Template Library*) n'est pas permise. Ce sera permis plus tard dans le cours.

Environnement de développement

Relisez les Politiques et les directives sur les outils informatiques dans le cours INF3105. Votre TP2 doit pouvoir être compilé avec g++ version 4.8.

Taille des équipes

Vous pouvez faire ce travail en équipe de 1 ou 2. Toutefois, tous les membres de l'équipe doivent contribuer à l'ensemble du travail et non à seulement quelques parties. Le travail d'équipe vise à favoriser les discussions et l'entraide. Le travail d'équipe ne vise pas à réduire la tâche. Ainsi, se diviser la tâche en deux n'est pas une méthode de travail d'équipe appropriée dans ce cours. La participation inadéquate des membres de l'équipe peut être considérée comme du plagiat. Le professeur et le correcteur pourront sélectionner quelques équipes au hasard afin de vérifier que tous les membres sont capables d'expliquer l'ensemble du travail.

Tests

Des tests sont disponibles dans tp2-tests.tar.bz2. La lettre en majuscule dans le nom des fichiers indique le critère de correction associé.

Aucun fichier test ne vérifie vraiment la limite `distanceTotalMax`. Il vous appartient de bien tester par vous-même.

Exécution des tests sur votre propre machine

Pour exécuter les tests, vous devez décompresser le fichier `tp2-tests.tar.bz2`. Les fichiers tests et le script `evaluer.sh` ne doivent pas être dans le même répertoire que votre programme `tp2`. Le script utilise un valideur qui doit être compilé une fois à partir du code source `valideur.cpp`.

./tests/evaluer.sh

```
...
Évaluation du TP2
...
Test            CPU     Mém.(k) Validation
exemple1.txt    0.00    3616k   27 bon(s)       0 erreur(s)     /27     |
        OK
exemple2.txt    0.00    3336k   20 bon(s)       0 erreur(s)     /20     |
        OK
exemple3.txt    0.00    3312k   19 bon(s)       0 erreur(s)     /19     |
        OK
exemple4.txt    0.00    3200k   12 bon(s)       0 erreur(s)     /12     |
        OK
exemple5.txt    0.00    3580k   15 bon(s)       0 erreur(s)     /15     |
        OK
testC0.txt      0.00    3344k   33 bon(s)       0 erreur(s)     /33     |
        OK
testC1.txt      0.00    3676k   303 bon(s)      0 erreur(s)     /303    |
        OK
testC2.txt      0.00    3624k   359 bon(s)      0 erreur(s)     /359    |
        OK
testC3.txt      0.02    4020k   364 bon(s)      0 erreur(s)     /364    |
        OK
testC4.txt      2.76    9432k   3435 bon(s)     0 erreur(s)     /3435   |
        OK
testC5.txt      38.17   16056k  30500 bon(s)    0 erreur(s)     /30500  |
        OK
testC6.txt      22.46   36752k  3251 bon(s)     0 erreur(s)     /3251   |
        OK
testD0.txt      0.00    3628k   402 bon(s)      0 erreur(s)     /402    |
        OK
testD1.txt      0.01    3700k   3110 bon(s)     0 erreur(s)     /3110   |
        OK
testD2.txt      0.56    4900k   6300 bon(s)     0 erreur(s)     /6300   |
        OK
testD3.txt      37.36   13180k  30600 bon(s)    0 erreur(s)     /30600  |
        OK
testE0.txt      0.00    3572k   402 bon(s)      0 erreur(s)     /402    |
        OK
testE1.txt      0.01    3680k   3110 bon(s)     0 erreur(s)     /3110   |
        OK
testE2.txt      390.48  4888k   6300 bon(s)     0 erreur(s)     /6300   |
        OK
...
```

Remise

Vous devez remettre électroniquement le TP2 **au plus tard le dimanche 27 mars 2022 à 23h59**. Votre projet, en format zip, doit être remis sur le site Moodle dans la section Travaux pratiques. Veuillez mettre vos codes permanents dans le nom de votre fichier.

Vous pouvez soumettre votre TP autant de fois que vous voulez. Seule la dernière soumission sera considérée.

Vous devez remettre tous vos fichiers sources, incluant un fichier Makefile et votre rapport.

Rapport

- **Auto-évaluation** indiquant si votre programme fonctionne correctement, partiellement ou aucunement.

- **Un tableau montrant les temps d'exécution sur les fichiers tests fournis.**
  Si un test prend plus de 60 secondes, vous pouvez l'arrêter ([Ctrl]+[C]) et simplement écrire **> 60** dans votre rapport.

  Vous **pourrez** générer un rapport automatiquement en lançant la commande /home/inf3105/tp2/evaluer.sh à partir du répertoire où vous avez compilé votre exécutable tp2.

- **Analyse de la complexité temporelle en notation grand O de chacune des commandes suivantes et de votre programme en général:**

  - PLACER;
  - APPROV;
  - RAMASSER; et
  - RECOMMANDER.

  Les complexités temporelles devraient être exprimées en fonction de :

  - *n* indique nombre d'épiceries;
  - *m* indique nombre de types de produit différents;
  - *k* indique nombre d'items sur la liste de la commande;
  - et de tout autre variable que vous jugeriez pertinent.

Évaluation

Ce travail pratique vaut 20% de la note finale.

Grille d'évaluation

Critère	Description	Pondération
A.	**Respect des directives pour la remise** • Fichiers sources (Makefile, .h, .cpp) seulement. Aucun fichier binaire (.o, exécutable). Aucun fichier test. • Remise adéquate par Oto. Pas de remise par courriel. • Compilable avec `make` sans modifications. • Respecte l'interface d'entrée/sortie..	/ 2

B.	**Appréciation générale**
	
	• Structure du programme + Qualité du code :
	  ◦ Choix des types de données; identificateurs (noms) significatifs, lisibilité du code, pertinence des commentaires; etc.
	  ◦ Justesse de l'usage du mot-clé `const`, des références (&) et des pointeurs (*).
	• Encapsulation :
	  ◦ Respect des principes de l'abstraction; utilisation appropriée de `public`, `private`, `friend`, etc.
	  ◦ Cachez le maximum de la représentation des objets en rendant un maximum d'attributs privés;
	  ◦ Évitez autant que possible les bris d'abstraction, comme des *getters* et *setters* qui retournent ou affectent directement des attributs d'un type abstrait de donnée. Par exemple, les fonctions `getLatitude()` et `getLongitude()` ne devraient pas exister dans une classe `PointST`. Mais, une fonction `getNom()` dans une classe `Épicerie` peut être justifiée.
	• Gestion de la mémoire :
	  ◦ Toute la mémoire allouée dynamiquement doit être correctement libérée au moment approprié et avant la fin du programme.
		/ 3

C.	**Fonctionnement de base** Les commandes RECOMMANDER auront nbMaxÉpicerie = 1. Les dates d'expiration peuvent être ignorées.	/ 3

D.	**Fonctionnement de base avec dates d'expiration.** Item C, excepté que les dates d'expiration doivent être considérées.	/ 3
E.	**Fonctionnement correct complet.** La majorité des tests seront avec des commandes RECOMMANDER ayant nbMaxÉpicerie = 1 à 3 épiceries.	/ 4
F.	**Efficacité** Votre programme doit utiliser judicieusement les arbres binaires de recherche.	/ 3
G.	**Analyse des algorithmes** Complexité temporelle des fonctions et justification.	/ 2
	Total :	/ 20
H.	**Boni : Efficacité avancée** 1. Jusqu'à 1 point boni peut être accordé si votre programme s'exécute **significativement** plus rapidement que les temps médians des programmes du groupe. 2. Un deuxième point boni peut être accordé à l'équipe qui aura les meilleurs temps d'exécution. Avertissements : 1. Pour être éligible aux points bonis, il faut obtenir 10/10 aux critère C, D et E. 2. Implémenter une version efficace peut parfois demander beaucoup de temps et être une source de bogues, et ce, pour seulement 1 point boni. 3. Il existe souvent un compromis entre l'efficacité d'un logiciel et ses autres qualités, comme le fonctionnement correct, la robustesse, la lisibilité du code, la maintenabilité, etc. Vous devez donc être prudents. Évitez les sacrifices qui pourraient vous pénaliser aux autres critères dont B, C, D et E.	+ 2
I.	**Pénalité : Utilisation de la STL** Un des objectifs du TP2 est d'implémenter l'arbre AVL tel que présenté en classe et dans les notes de cours. À noter que les Lab6 et Lab7 viseront à compléter l'implémentation d'un arbre AVL et d'un dictionnaire (ArbreMap). Si vous éprouvez des difficultés à réaliser votre propre implémentation d'arbre AVL, il est permis d'utiliser les	- 3

	conteneurs de la librairie standard de C++ (la STL), dont les classes `set` et `map`. Toutefois, une pénalité de 3 points sera appliquée.	
	Note maximale :	22 / 20

* Bien que les critères C, D et E visent le fonctionnement correct en premier lieu, l'efficacité peut être indirectement évaluée lorsqu'un programme ne parvient pas à produire des résultats dans des délais raisonnables.

Pour les cas problématiques, jusqu'à 3 points peuvent être retranchés pour la qualité de la langue ou de la présentation.

Autres exemples

Exemple 2

```
./tp2 < exemple2.txt > exemple2+.txt
```

exemple2.txt	exemple2+.txt
`PLACER Foudici (45.506873,-73.568921) ;`	`OK`
`APPROV Foudici :`	
`  Fraises 1 2017-10-29`	
`  Fraises 2 2017-10-27 ;`	`OK`
`APPROV Foudici :`	
`  Fraises 3 2017-10-28`	
`  Fraises 4 2017-10-30 ;`	`OK`
`DATE 2017-10-26 ;`	`OK`
`INVENTAIRE Foudici ;`	`Fraises 10 ;`
`DATE 2017-10-27 ;`	`OK`
`INVENTAIRE Foudici ;`	`Fraises 8 ;`
`DATE 2017-10-28 ;`	`OK`
`INVENTAIRE Foudici ;`	`Fraises 5 ;`
`DATE 2017-10-29 ;`	`OK`
`INVENTAIRE Foudici ;`	`Fraises 4 ;`
`APPROV Foudici :`	
`  Fraises 4 2017-10-30 ;`	`OK`
`DATE 2017-10-30 ;`	`OK`
`INVENTAIRE Foudici ;`	`;`
`APPROV Foudici :`	
`  Fraises 4 2017-11-01 ;`	`OK`
`INVENTAIRE Foudici ;`	`Fraises 4 ;`
`DATE 2017-11-01 ;`	`OK`
`INVENTAIRE Foudici ;`	`;`
`DATE 2017-11-02 ;`	`OK`
`INVENTAIRE Foudici ;`	`;`

Exemple 3

```
./tp2 < exemple3.txt > exemple3+.txt
```

exemple3.txt	exemple3+.txt
PLACER Foudici (45.506873,-73.568921) ; PLACER IGA (45.507798,-73.563369) ; APPROV Foudici : Pommes 4 2099-12-31 ; APPROV IGA : Pommes 3 2099-12-31 ; RAMASSER Pommes 1 ; Foudici ; INVENTAIRE Foudici ; RAMASSER Pommes 1 ; IGA ; INVENTAIRE IGA ; APPROV IGA : Fraises 3 2099-12-31 ; INVENTAIRE IGA ; RAMASSER Pommes 1 Fraises 1 ; Foudici IGA ; INVENTAIRE IGA ; INVENTAIRE Foudici ; RAMASSER Pommes 1 Fraises 1 ; IGA Foudici ; INVENTAIRE IGA ; INVENTAIRE Foudici ; RAMASSER Pommes 2 Fraises 2 ; IGA Foudici ; INVENTAIRE IGA ; INVENTAIRE Foudici ;	OK OK OK OK COMPLET Pommes 3 ; COMPLET Pommes 2 ; OK Fraises 3 Pommes 2 ; COMPLET Fraises 2 Pommes 2 ; Pommes 2 ; COMPLET Fraises 1 Pommes 1 ; Pommes 2 ; MANQUE Fraises 1 ; ; Pommes 1 ;

Exemple 4

```
./tp2 < exemple4.txt > exemple4+.txt
```

exemple4.txt	exemple4+.txt
PLACER Foudici (45.506873,-73.568921) ; DATE 2017-11-01 ; APPROV Foudici : Pommes 2 2017-11-05 Pommes 3 2017-11-11 Pommes 7 2017-11-17 ; INVENTAIRE Foudici ; RAMASSER Pommes 1 ; Foudici ; INVENTAIRE Foudici ; DATE 2017-11-06 ; INVENTAIRE Foudici ; DATE 2017-11-15 ; INVENTAIRE Foudici ; DATE 2017-11-20 ; INVENTAIRE Foudici ;	OK OK OK Pommes 12 ; COMPLET Pommes 11 ; OK Pommes 9 ; OK Pommes 6 ; OK ;

Exemple 5

```
./tp2 < exemple5.txt > exemple5+.txt
```

exemple5.txt	exemple5+.txt
DATE 2017-11-01 ; PLACER Foudici (45.506873,-73.568921) ; PLACER IGA (45.507798,-73.563369) ; APPROV Foudici : Pommes 4 2017-11-20 ; APPROV IGA : Pommes 3 2017-11-14 Fraises 3 2017-11-27 ; RAMASSER Pommes 2 Fraises 1 ; IGA Foudici ; INVENTAIRE IGA ; INVENTAIRE Foudici ; RAMASSER Pommes 1 Fraises 1 ; Foudici IGA ; INVENTAIRE IGA ; INVENTAIRE Foudici ; RECOMMANDER (45.509339,-73.568465) 1 999000 : Fraises 1 ; RECOMMANDER (45.509339,-73.568465) 1 999000 : Fraises 2 ; RECOMMANDER (45.509339,-73.568465) 1 999000 : Fraises 1 Pommes 2 ; RECOMMANDER (45.509339,-73.568465) 2 999000 : Fraises 1 Pommes 2 ;	OK OK OK OK OK COMPLET Fraises 2 Pommes 1 ; Pommes 4 ; COMPLET Fraises 1 Pommes 1 ; Pommes 3 ; 865m IGA IMPOSSIBLE IMPOSSIBLE 1154m Foudici IGA