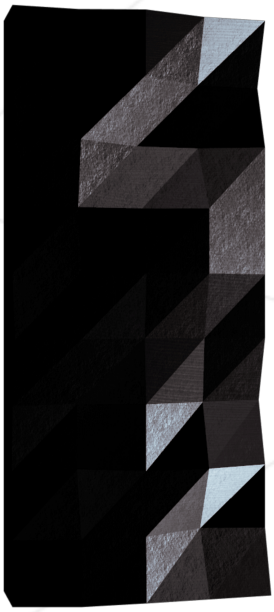


Projet d'Algorithmes Elémentaires

lem-in

42 staff staff@42.fr

Résumé: Ce projet a pour but de vous faire coder un gestionaire de fourmillière.

Table des matières

Chapitre I

Préambule

Voici un extrait de Bill & John

```
Up to mighty London
Came an Irishman one day.
As the streets are paved with gold
Sure, everyone was gay,
Singing songs of Piccadilly,
Strand and Leicester Square,
Till Paddy got excited,
Then he shouted to them there:
It's a long way to Tipperary,
It's a long way to go.
It's a long way to Tipperary
To the sweetest girl I know!
Goodbye, Piccadilly,
Farewell, Leicester Square!
It's a long long way to Tipperary,
But my heart's right there.
(repeat)
Paddy wrote a letter
To his Irish Molly-O,
Saying, "Should you not receive it,
Write and let me know!"
"If I make mistakes in spelling,
Molly, dear," said he,
"Remember, it's the pen that's bad,
Don't lay the blame on me!
It's a long way to Tipperary,
It's a long way to go.
It's a long way to Tipperary
To the sweetest girl I know!
Goodbye, Piccadilly,
Farewell, Leicester Square!
It's a long long way to Tipperary,
But my heart's right there.
```

```
Molly wrote a neat reply
To Irish Paddy-O,
Saying "Mike Maloney
Wants to marry me, and so
Leave the Strand and Piccadilly
Or you'll be to blame,
For love has fairly drove me silly:
Hoping you're the same!"
It's a long way to Tipperary,
It's a long way to go.
It's a long way to Tipperary
To the sweetest girl I know!
Goodbye, Piccadilly,
Farewell, Leicester Square!
It's a long long way to Tipperary,
But my heart's right there.
```

Ce projet est beaucoup plus facile si vous le réalisez en regardant `Bill & John`.

Chapitre II

Sujet - Partie obligatoire

II.1 Introduction

Des étudiants en magie ont fabriqué Hex, une machine à penser. À base de fourmis pour les calculs, de ruches et d'abeilles pour la mémoire, d'une souris pour (...euh oui pourquoi au fait ?), de fromage (pour nourrir la souris), d'une plume pour écrire.

Pour plus d'info réferez-vous aux livres** de Terry Pratchett, histoire de changer du 42. (**) Ook !

On va s'intéresser plus particulièrement à sa partie calculs. Son fonctionnement ? Simple ! On monte une fourmilière avec tout son lot de tunnels et de salles, on met des fourmis d'un côté et on regarde comment elles trouvent la sortie.
Comment on monte une fourmilière ? On a besoin de tubes et de boîtes.
On relie les boîtes entre elles par plus ou moins de tubes. Un tube relie deux boîtes entre elles et pas plus.
Une boîte peut être reliée à une infinité d'autres boîtes par autant de tubes qu'il en faudra.
Ensuite on enterre le tout (où vous voudrez) pour que les fourmis ne puissent pas tricher en regardant avant de commencer.
Comme ici, on est pas trop bricolage à coup de boîtes, scotch et bouts de ficelle, on va en faire une version high-tech.
Le but du projet sera donc de faire un simulateur de "Hex".

II.2 Comment ça marche ?

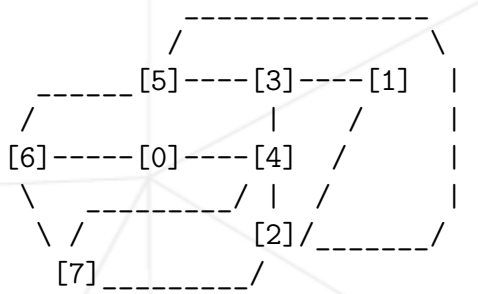
- Votre programme va recevoir sur l'entrée standard la description de la fourmilière pour la forme suivante :

```
nombre_de_fourmis
les_salles
les_tubes
```

- La fourmilière est décrite par ses liaisons :

```
##start
1 23 3
2 16 7
#commentaire
3 16 3
4 16 5
5 9 3
6 1 5
7 4 8
##end
0 9 5
0-4
0-6
1-3
4-3
5-2
3-5
#autre commentaire
4-2
2-1
7-6
7-2
7-4
6-5
```

- Ce qui représente :



- On a donc 2 parties :
  - La définition des salles sous la forme suivante : `nom coord_x coord_y`
  - La définition des tubes : `nom1-nom2`
  - Le tout entre-coupé de commentaires qui commencent par `#`

`Les noms des salles ne seront pas forcément des chiffres, ils ne seront pas forcément dans l'ordre, et encore moins continus (on pourra trouver des salles aux noms zdfg, 256, qwerty, etc ...). Mais une salle ne commencera jamais par le caractère L ou le caractère #`

`Les coordonnées des salles seront toujours des nombres entiers.`

- Les lignes commençant par un `##` sont des commandes modifiant les propriétés de la ligne significative qui vient juste après.
- Par exemple, `##start` indique l'entrée de la fourmilière et `##end` la sortie.

`Toute commande inconnue sera ignorée.`

- La première ligne non conforme ou vide entraîne la fin de l'acquisition de la fourmilière et son traitement normal avec les données déjà acquises.
- S'il n'y a pas assez de donnée pour faire un traitement normal vous devrez simplement afficher `ERROR`

II.3 Le but de tout ca ?

- Le but du projet est de trouver le moyen le plus rapide de faire traverser la fourmilière par `n` fourmis.
- Évidemment, il y a quelques contraintes. Pour arriver le premier, il faut prendre le chemin le plus court (et pas forcément pour autant le plus simple), ne pas marcher sur ses congénères, tout en évitant les embouteillages.
- Au début du jeu, toutes les fourmis sont sur la salle indiquée par la commande `##start`. Le but est de les amener sur la salle indiquée par la commande `##end` en prenant le moins de tours possible. Chaque salle peut contenir une seule fourmi à la fois (sauf `##start` et `##end` qui peuvent en contenir autant qu'il faut).
- On considère que les fourmis sont toutes dans la salle `##start` au démarrage
- Vous n'afficherez à chaque tour que les fourmis qui ont bougé
- À chaque tour vous pouvez déplacer chaque fourmi une seule fois et ce suivant un tube (la salle réceptrice doit être libre).
- Vous devez sortir le résultat sur la sortie standard sous la forme suivante :

```
nombre_de_fourmis
les_salles
les_tubes

Lx-y Lz-w Lr-o ...
```

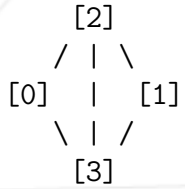
Où `x, z, r` sont des numéros de fourmis (allant de 1 à `nombre_de_fourmis`) et `y, w, o` des noms de salles.

- Exemple 1 :
`[0]-[2]-[3]-[1]`

```
zaz@blackjack /tmp/lem-in $ ./lem-in < sujet1.map
3
##start
0 1 0
##end
1 5 0
2 9 0
3 13 0
0-2
2-3
3-1

L1-2
L1-3 L2-2
L1-1 L2-3 L3-2
L2-1 L3-3
L3-1
zaz@blackjack /tmp/lem-in $
```

- Exemple 2 :




```
zaz@blackjack /tmp/lem-in $ ./lem-in < sujet2.map
3
2 5 0
##start
0 1 2
##end
1 9 2
3 5 4
0-2
0-3
2-1
3-1
2-3

L1-3 L2-2
L1-1 L2-1 L3-3
L3-1
zaz@blackjack /tmp/lem-in $
```

Ce n'est pas aussi simple que ça en a l'air.

- Quant à savoir quel type d'opération les étudiants en magie pouvaient bien faire avec un tel ordinateur, tout ce qu'on en sait aujourd'hui, c'est que l'électricité c'est vachement plus fiable.

Chapitre III

Sujet - Partie bonus

Les bonus ne seront évalués que si votre partie obligatoire est EXCELLENTE. On entend par là qu'elle est entièrement réalisée, que votre gestion d'erreur est au point, même dans des cas vicieux, ou des cas de mauvaise utilisation. Concrètement, si votre partie obligatoire n'est pas parfaite, vos bonus seront intégralement IGNORÉS.

Voici quelques idées de bonus intéressants à réaliser, voire même utiles. Vous pouvez évidemment ajouter des bonus de votre invention, qui seront évalués à la discrétion de vos correcteurs.

- En bonus, pourquoi ne pas coder un visualiseur de fourmilière ?
  - Soit en deux dimensions, vue de "dessus". Voir pourquoi pas du point de vue d'une fourmi dans un environnement de couloirs en 3D.
  - Pour l'utiliser il suffirait d'un `./lem-in < map_fourmilliere.txt | ./visu-hex`
  - À noter que puisque les commandes et commentaires sont répétés sur la sortie, il est donc possible de passer des commandes spécifiques au visualisateur (pourquoi pas des couleurs, des niveaux ?)
  - Vous l'aurez noté, les coordonnées des salles seront utiles seulement ici.

Chapitre IV

Consignes

- Ce projet sera corrigé par des humains. Vous êtes donc libres d'organiser et nommer vos fichiers comme vous le désirez, en respectant néanmoins les contraintes du sujet.
- L'exécutable doit s'appeller `lem-in`.
- Votre Makefile devra compiler le projet(et ses bonus dans le cas échéant), et doit contenir les règles habituelles. Il ne doit recompiler le programme qu'en cas de nécessité.
- Votre projet doit être à la Norme.
- Vous devez gérer les erreurs de façon sensible. En aucun cas votre programme ne doit quitter de façon inattendue (Segmentation fault, etc...)
- Vous devez rendre, à la racine de votre dépôt de rendu, un fichier `auteur` contenant votre login suivi de '\n' :

```
$>cat -e auteur
xlogin$
$>
```

- Dans le cadre de votre partie obligatoire, vous avez le droit d'utiliser les fonctions suivantes :
  - read
  - write
  - malloc
  - free
  - perror
  - strerror
  - exit
- Vous avez l'autorisation d'utiliser ce que bon vous semble pour votre visualiseur (si vous en faites un)
- Vous pouvez poser vos questions sur le forum, sur jabber, Slack...