# Devoir maison

## Recherche opérationnelle S3.

Afizullah RAHMANY
Numéro d'étudiant : 20170171

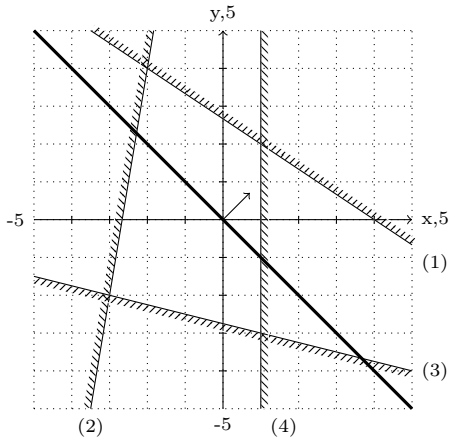
*Ce devoir maison a été généré automatiquement et aléatoirement pour Afizullah RAHMANY. Il contient 3 exercices : un exercice d'application du cours sur 1 point, un exercice intermédiaire sur 1 point et un algorithme à coder sur 2 points. Pour répondre à ce devoir, vous devez rendre, sur* `exam.ensiie.fr`*, un fichier texte nommé* `rahmanyafizullah_20170171.txt`*. Ce fichier devra contenir* ***dans l'ordre*** *les solutions des 3 exercices tels que demandé dans chaque exercice. Toute ligne vide dans le fichier sera ignorée (vous pouvez donc sauter des lignes entre 2 exercices, ou au sein d'un exercice). ATTENTION : un non respect du format pour un exercice entraînera la note de 0 pour l'exercice. Il vous est possible de tester votre format ainsi que votre dernier exercice avec le script* `check` *qui vous a été fourni avec ce sujet. Vous pouvez par exemple le copier avec votre fichier solution sur une machine de l'école et exécuter*

```
./check rahmanyafizullah_20170171.txt
```

*Si votre fichier est au bon format, le script vous le dira explicitement. Le code de votre dernier exercice peut être dans ce cas testé sur 100 tests unitaires. Si un test ne passe pas, le script vous l'affichera pour que vous puissiez corriger votre code. Vous pouvez demander de l'aide aux professeur/chargés de TD, ou aux autres élèves, mais sachez que le sujet est différent pour tout le monde.*

**Exercice** — ***#1-5-1 Programme linéaire à partir de la version graphique***

Un programme linéaire à 2 variables donne la représentation graphique ci-dessous. La fonction de coût est en gras, et la flèche indique dans quel sens elle augmente. Pour chaque contrainte, le demi plan du côté des hachures indiquent quel est le côté non réalisable. On souhaite minimiser. Chaque droite passe nécessairement par 2 points de coordonnées entières.




A partir du dessin ci-dessus, déterminez le programme linéaire sous forme matricielle.

Vous donnerez cette réponse au format suivant : vous noterez en premier le numéro de l'exercice précédé par un `#`,
puis 1 ligne contenant 2 entiers séparés par une espace : le coût de $x$ et le coût de $y$ ; puis 4 lignes avec 3 entiers $a$, $b$ et $c$ séparés par des espaces représentant chaque contrainte $a \cdot x + b \cdot y \leq c$ du programme. Les entiers de chaque ligne (y compris la première) doivent être premiers entre eux. Les contraintes doivent être données dans l'ordre indiqué sur le dessin.

Par exemple :

```
# 1-5-1
-2 1
```

```
0 1 2
1 -3 2
...
```

**Exercice** — ***#2-7-3 Taille moyenne d'une file d'attente : taux de naissance variable***

On considère une file d'attente à 1 guichet avec un taux de naissance non constant et un taux de mort constant $\mu = 6$. Le taux de naissance vaut $\lambda_n = 3$ si $n < 5$ et $\lambda_n = 3$ sinon. Calculez le nombre moyen de personnes dans la file en régime permanent, si ce régime existe.

Vous donnerez cette réponse au format suivant : vous noterez en premier le numéro de l'exercice précédé par un `#`,
puis une ligne contenant un flottant égal à la taille moyenne de la file si le régime permanent existe. Ce nombre doit être écrit avec la notation scientifique, arrondi 2 chiffres après la virgules. Si le régime permanent n'existe pas alors la ligne doit contenir à la place le nombre `-1`.

Par exemple :

```
# 2-7-3
1.20E+1
```

**Exercice** — ***#3-6-4 Algorithme de calcul des états transitoires***

Coder en C un algorithme calculant les états transitoires d'une chaîne de Markov à partir de ses classes d'états communiquants.

Pour cela, vous coderez une fonction avec la signature suivante :

`void transient_states_algorithm(int n, float** transitions, int* comclasses, int* transientstates);`

- `n` est le nombre d'états de la chaîne de Markov numérotés de 1 à $n$.
- `transitions` est un tableau de `n` × `n` flottants où `transitions[i-1][j-1]` est la probabilité de transition de l'état $i$ vers l'état $j$.
- `comclasses` est un tableau de `n` entiers et représentent les classes d'états communicants. Deux étatts $i$ et $j$ sont dans la même classe si et seulement si `comclasses[i-1]` = `comclasses[j-1]`.
- `transientstates` est un tableau de `n` entiers. En entrée, `transientstates[i-1]` vaut -1 pour tout $1 \leq i \leq n$. Votre fonction doit affecter dans `transientstates[i-1]` l'entier 1 si l'état $i$ est un état transitoire et 0 sinon.

On garantit, que, quelque soit l'entrée $2 \leq n \leq 100$. Le tableau `comclasses` contient tous les entiers entre 1 et le nombre de classes d'états communiquants.

Votre fonction doit compiler avec `gcc -Wall -Wextra -ansi` et se terminer en moins de 1 seconde. Vous pouvez déclarer d'autres fonctions ou des structures mais ne pouvez ni écrire de `#define` ou de `#include`. Vous pouvez utiliser tout ce qui est défini dans `stdlib.h` à l'exception de `system`. Cette bibliothèque sera inclue automatiquement. Ne mettez pas de commentaire. Elle sera testée sur 100 cas. Votre note est de 2 si tous les tests sont réussis, 1 si plus de 10 pourcents mais pas tous, et 0 sinon. **Conseil** : n'utilisez pas de tableaux, utilisez des pointeurs à la place.

Vous donnerez cette réponse au format suivant : vous noterez en premier le numéro de l'exercice précédé par un `#`, puis écrivez votre code.

Par exemple :

```
# 3-6-4
void transient_states_algorithm(int n, float** transitions, int* comclasses, int* transientstates)
}
```