OI 2010 **Finale** 12 Mai 2010	**Remplissez ce cadre en MAJUSCULES et LISIBLEMENT, svp** PRÉNOM NOM : ..... ÉCOLE : ..... SALLE : CANDIX / DAO MACHINE N°	**Réservé**

Olympiades belges d'Informatique (durée : 1h15 maximum)

Ce document est le questionnaire de **la partie machine** de la finale des Olympiades belges d'Informatique pour **la catégorie secondaire**. Il comporte une question qui doit être résolue en **1h15 au maximum**. Seul le code produit par le participant sera pris en compte pour l'évaluation de cette partie.

Délivrables

1. Votre programme doit pouvoir être exécuté par la commande `run` prenant deux paramètres : le chemin vers fichier d'entrée et le chemin vers le fichier de sortie. Tous les détails sont donnés à la page suivante.
2. Vous devez également rendre votre questionnaire, avec le cadre en haut de première page correctement complété.

Notes générales (à lire attentivement avant de répondre à la question)

1. Indiquez votre nom, prénom, école, **nom de la salle et numéro de la machine** sur la première page. Posez votre carte d'étudiant ou carte d'identité sur la table.
2. Installez-vous à la **place** qui vous a été **attribuée** par les organisateurs.
3. Vous ne pouvez avoir que de quoi écrire avec vous, les calculatrices, GSM, ... sont **interdits**. Laissez toutes vos affaires à l'endroit indiqué par les surveillants, ne prenez que de quoi écrire avec vous.
4. Vous **ne pouvez** à aucun moment **communiquer** avec qui que ce soit, excepté avec les surveillants ou les organisateurs. Toute question portant sur la compréhension de la question ou liée à des problèmes techniques ne peut être posée qu'aux organisateurs. Toute question logistique peut être posée aux surveillants.
5. Vous **pouvez** utiliser toutes les fonctionnalités de la librairie standard du langage que vous aurez choisi.
6. Vous pouvez demander des **feuilles de brouillon** aux surveillants.
7. Il est strictement **interdit de manger ou boire** dans les salles informatiques.
8. Les participants **ne peuvent en aucun cas quitter leur place** pendant l'épreuve, par exemple pour aller aux toilettes ou pour fumer une cigarette.
9. Vous avez **exactement une heure et quart** pour résoudre le problème.

Bonne chance !

Questionnaire finale machine secondaire

OI 2010 Mercredi 12 mai 2010	**Réservé**

Instructions pratiques

Étape 1 – Récupérer le squelette

- Ouvrez le répertoire `OI2010` sur le bureau ainsi que les répertoires `skeleton` et `prog` qu'il contient.
- Dans `skeleton`, sélectionnez le langage que vous souhaitez utiliser. Copiez le **contenu** du répertoire de ce langage dans le répertoire `prog`.

Étape 2 – Tester le squelette

- Lancez un terminal (menu `Applications > Outils Système > Terminal`) et déplacez vous dans le répertoire de votre programme en tapant dans la ligne de commande `cd Bureau/OI2010/prog`.
- Tapez `make` dans le terminal, dans le répertoire courant. Ceci permet de compiler votre programme (quand c'est nécessaire) et génére un exécutable nommé `run`, commun à tous les langages. C'est ce dernier qui sera utilisé pour lancer les tests et vous évaluer.
- Tapez `test_sec --all` dans un terminal pour exécuter les tests automatiques. Ceci permet de tester le code contenu dans le répertoire `prog` sur quelques exemples simples et affichera un compte rendu. Ces tests **ne sont PAS** ceux qui seront exécutés pour évaluer votre code, mais l'évaluation de votre code par les examinateurs fonctionnera exactement de la même façon. **Il est donc primordial que ces tests fonctionnent à la fin de l'épreuve.** Si ce n'est pas le cas, nous ne pourrons vous attribuer le moindre point.

Étape 3 – Modifier le programme

- Vous pouvez maintenant modifier le programme contenu dans le répertoire `prog`.
- Après chaque modification, ré-exécutez `make` (vous devez être dans le répertoire `prog` pour pouvoir faire ça, et pour y revenir à tout moment, tapez `cd ~/Bureau/OI2010/prog`).

Étape 4 – Tester votre programme

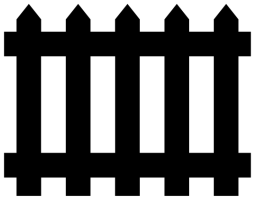
- Lorsque `make` a été exécuté et qu'aucune erreur n'a été détectée, un fichier `run` est créé. Vous pouvez dès lors tester votre programme en exécutant (dans le répertoire `prog`) :
  ```
  run fichier_d_entree fichier_de_sortie
  ```
- Vous pouvez utiliser les quelques fichiers d'entrée qui vous sont donnés dans le répertoire `OI2010/tests`, par exemple :
  ```
  run ../tests/test1 out
  ```
  Après exécution de cette commande, le fichier `out` contient la sortie de votre programme.
- À tout moment, lorsque votre programme fonctionne, exécutez les tests automatique en lançant la commande `test_sec --all`.

Remarques

- Les documentations pour chaque langage se trouvent dans le répertoire `OI2010/doc`.
- Ne prenez pas le risque de vous retrouver à la fin de l'heure avec un programme qui ne fonctionne plus, alors qu'il fonctionnait auparavant ! Faites des backups (faites des copie de votre répertoire `prog`) chaque fois que votre programme a été amélioré et fonctionne. Vous pourrez ainsi récupérer votre code si nécessaire. Utilisez par exemple le répertoire `OI2010/backup` et n'hésitez pas à conserver différentes versions de votre programme.
- Si vous avez besoin d'aide, sur certains points de cette feuille, demandez de l'aide à un surveillant.

Vous n'avez qu'une heure et quart pour cette épreuve. Préférez une solution peu performante qui fonctionne à une solution ambitieuse qui ne fonctionne finalement pas !

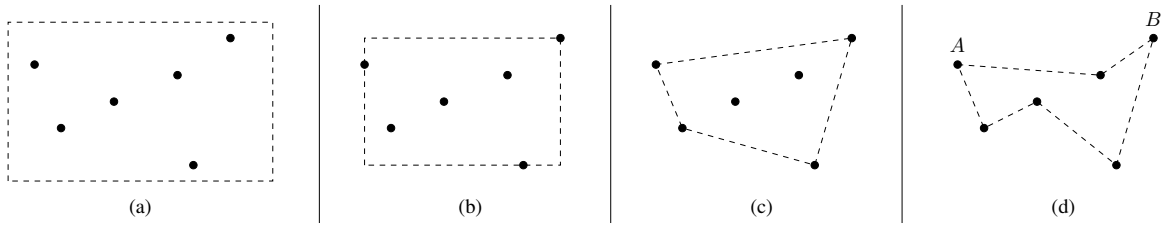
Problème de clôture



Vous avez récemment hérité d'un grand verger comportant un certain nombre d'arbres. Afin d'éviter que des personnes malveillantes s'introduisent sur votre terrain et saccagent le verger ou volent des fruits, vous souhaitez poser une clôture tout autour du verger. Néanmoins, ce type de matériel coute assez cher et, avec la crise actuelle, il vous faut absolument minimiser la longueur de clôture que vous allez acheter.

La clôture consiste en un certain nombre de poteaux le long desquels vous allez tendre la clôture, l'enclos ainsi formé devant être un polygone convexe. Votre ami peut vous fournir gratuitement autant de poteaux que vous le souhaitez. Il ne vous reste plus qu'à choisir comment placer votre clôture afin de minimiser sa longueur et faire en sorte que tous les arbres se trouvent dans l'enclos. Votre unique contrainte est que, afin de faciliter les déplacements dans l'enclos, celui-ci doit être convexe, c'est-à-dire qu'on doit pouvoir se déplacer entre deux points quelconques de l'enclos, en ligne droite directe, sans jamais devoir le quitter ou devoir escalader une clôture.

Comme l'illustre la figure suivante montrant un verger de six arbres, il y a plusieurs solutions possibles.



Les solutions (a), (b) et (c) sont toutes les trois acceptables, et sont de plus en plus optimales. La troisième étant meilleure que la première, elle vous rapportera plus de points. Par contre, la solution (d) n'est pas acceptable étant donné que le polygone formé n'est pas convexe. En effet, si on part du point A, il n'est pas possible de rejoindre le point B en ligne droite tout en restant dans l'enclos.

Tâche

Écrire un programme qui, étant donné N arbres de coordonnées cartésiennes entières positives données, calcule les coordonnées cartésiennes des poteaux à placer afin de poser une clôture qui contient tous les arbres et qui forme un polygone convexe. Pour avoir le score maximum, vous devez minimiser le périmètre de l'enclos ainsi défini.

Limites et contraintes

Votre programme ne doit pouvoir gérer que les problèmes se situant dans ces limites. Tous les tests effectués resteront dans ces limites.

- $3 \leq N \leq 100\,000$
- $0 \leq X, Y \leq 20\,000$ pour toutes les coordonnées cartésiennes des arbres et des poteaux

Dans les ensembles de données qui seront utilisés par les examinateurs pour tester vos programmes :

- Les arbres ne sont pas tous colinéaires, c'est-à-dire qu'ils ne se trouvent pas tous sur une même droite ;
- Il n'y a pas deux arbres se situant au même endroit.

Dans le résultat produit par votre programme :

- Les poteaux peuvent être placés à la même coordonnée qu'un arbre, mais vous ne pouvez pas placer plusieurs poteaux à la même coordonnée ;
- L'enclos délimité par vos poteaux doit être convexe ;
- Les coordonnées cartésiennes des poteaux doivent être fournies dans l'ordre dans lequel il faut les relier et de manière à ce que la clôture ainsi formée soit parcourue dans le **sens anti-horloger**.

Entrée

Le fichier d'entrée fourni est composé comme suit :

- La première ligne comporte un entier positif N : le nombre d'arbres ;
- Les N lignes suivantes comportent les coordonnées cartésiennes des arbres sous la forme de deux entiers positifs, l'abscisse et l'ordonnée, séparés par un espace unique.

Le fichier se termine par un saut de ligne.

Sortie

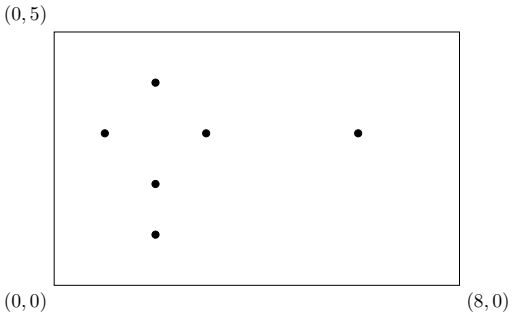
Le fichier de sortie à produire définit les coordonnées cartésiennes des poteaux de clôture à installer. Chaque coordonnée est définie sur une ligne du fichier, composée de deux entiers positifs, l'abscisse et l'ordonnée, séparés par un espace unique. Pour rappel, les coordonnées doivent être données dans l'ordre anti-horloger. Le fichier doit se terminer par un saut de ligne.

Exemple

Soit le fichier d'entrée `input.txt` suivant (et le verger qu'il représente) :

`input.txt`

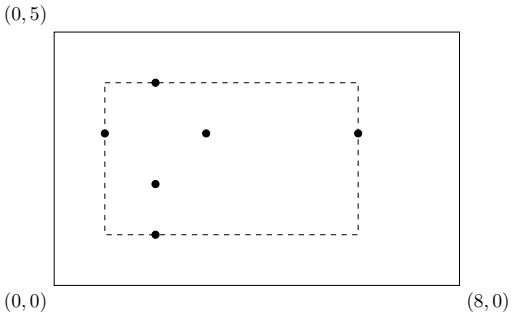
```
6
1 3
2 1
2 2
2 4
3 3
6 3
```



Une solution possible, obtenue après exécution de la commande `run input.txt output.txt` (et la clôture qu'elle représente) est donnée ci-dessous :

`output.txt`

```
1 4
1 1
6 1
6 4
```



Astuce

Pour tester si un point r de coordonnées cartésiennes (r_x, r_y) se trouve à gauche de la droite dirigée $\overrightarrow{pq}$ avec p de coordonnées (p_x, p_y) et q de coordonnées (q_x, q_y), il suffit de vérifier que la valeur de $(q_x r_y + p_x q_y + p_y r_x) - (q_x p_y + r_x q_y + r_y p_x)$ est positive.

Score

Toute clôture non acceptable (par exemple non convexe ou ne comprenant pas tous les arbres) ne donnera aucun points. Afin de calculer votre score, votre algorithme sera exécuté sur de nombreux fichiers d'entrée différents. Si le résultat est correct, des points seront attribués selon le périmètre de la clôture, un périmètre plus petit rapportant plus de points. Dans tous les cas, votre algorithme sera **arrêté après dix secondes**.

- 80% des points seront attribués sur des fichiers contenant moins de 1 000 arbres ;
- 20% des points seront attribués sur des fichiers de plus de 1 000 arbres.