

<div style="border: 2px solid black; padding: 5px; display: inline-block;">OI 2010</div> 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 supérieur**. 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 supérieur

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 génère 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_sup --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_sup --all`.

Remarques

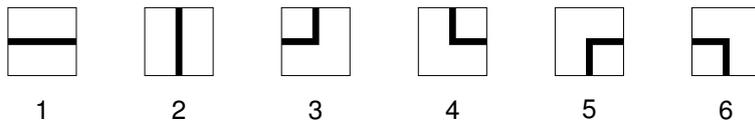
- Les documentations pour chaque langage se trouvent dans le répertoire `OI2010/doc`.
- Ne prenez pas le risque de vous retrouver à la fin de l'heure avec un programme qui ne fonctionne plus, alors qu'il fonctionnait auparavant ! Faites des backups (faites des copie de votre répertoire `prog`) chaque 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 !

À la conquête du monde

Vous avez hérité d'un immense terrain et souhaitez y bâtir une ville. Étant donné vos nombreux ennemis, vous souhaitez également construire une grande et longue muraille. Pour cela, vous avez lancé un appel auprès des architectes du monde entier.

Tous les différents projets soumis par les architectes peuvent se classer en six catégories, en fonction du type de muraille proposé. Ces six possibilités sont reprises ci-dessous :



Vous devez maintenant décider quels sont les projets que vous souhaitez accepter, et où ils vont être construits sur votre terrain. Ce que vous désirez, afin de protéger votre ville, est d'avoir, parmi tous les différents murs qui seront bâtis, le mur le plus long possible. Pour chaque projet, vous avez deux possibilités : l'accepter et l'affecter à une coordonnée sur votre terrain sur laquelle aucun projet n'a encore été attribué ou rejeter le projet.

Tâche

Écrire un programme qui, étant donné un terrain de C colonnes et L lignes et une liste de N projets, produit une séquence d'ordre (rejeter ou accepter le projet et établir les coordonnées où le construire) afin d'obtenir le plus long mur possible d'un seul tenant, parmi tous les murs ainsi définis.

Limites

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

- $1 \leq C, L \leq 1\,000$
- $1 \leq N \leq 1\,000\,000$

Entrée

Le fichier d'entrée fourni est composé comme suit :

- La première ligne comporte trois entiers positifs séparés par un espace unique : C et L , les dimensions du terrain et N le nombre de projets à examiner ;
- Chacune des N lignes suivantes comporte un entier positif dont la valeur vaut 1, 2, 3, 4, 5 ou 6 selon le type de muraille du projet (voir figure en début de page).

Vous ne pouvez faire aucune supposition quant à la répartition des types de muraille des projets (par exemple, vous pourriez avoir un fichier avec $S = 500$ projets, tous de type 2).

Sortie

Le fichier de sortie à produire décrit la séquence d'actions qu'il faut entreprendre. Deux types d'actions sont possibles :

- R indique qu'un projet est rejeté ;
- A X Y indique qu'un projet est accepté et placé en position (X, Y) sur le terrain.

X et Y sont des entiers strictement positifs représentant les coordonnées de l'endroit où placer le projet. X est le numéro de la colonne et Y est le numéro de la ligne. Le coin supérieur gauche est donc à la coordonnée (1, 1) et le coin inférieur droit à la coordonnée (C, L). La i^e ligne du fichier de sortie correspond à l'action qui sera appliquée pour le projet se situant à la $(i + 1)^e$ ligne du fichier d'entrée.

Exemple

Soit le fichier d'entrée suivant :

```
input.txt
5 3 6
1
5
1
2
3
2
```

Ce fichier indique donc que le terrain possède $C = 5$ colonnes et $L = 3$ lignes. Le tas contient $N = 6$ pièces qui sont, dans l'ordre, les pièces de numéro 1, 5, 1, 2, 3 et 2, c'est-à-dire  ,  ,  ,  ,  et  .

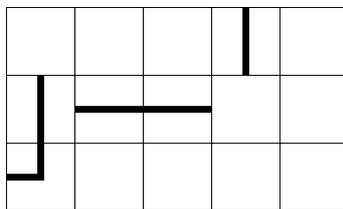
Une solution possible, obtenue après exécution de la commande `run input.txt output.txt`, est donnée ci-dessous :

```
output.txt
A 2 2
R
A 3 2
A 1 2
A 1 3
A 4 1
```

Voici en détails à quelles actions correspond ce fichier :

Pièce	Action	
	A 2 2	Le premier projet est placée en (2, 2), c'est-à-dire à la 2 ^e colonne, 2 ^e ligne
	R	Le second projet est jeté
	A 3 2	Le troisième projet est placée en (3, 2), c'est-à-dire à la 3 ^e colonne, 2 ^e ligne
	A 1 2	Le quatrième projet est placée en (1, 2), c'est-à-dire à la 1 ^{re} colonne, 2 ^e ligne
	A 1 3	Le quatrième projet est placée en (1, 3), c'est-à-dire à la 1 ^{re} colonne, 3 ^e ligne
	A 4 1	Le sixième projet est placée en (4, 1), c'est-à-dire à la 4 ^e colonne, 1 ^{re} ligne

Exécuter ces actions produit la grille suivante. La longueur du plus long mur ainsi obtenu est donc de deux.



Remarques

On ne peut placer un nouveau projet en (X, Y) que si aucun autre projet n'a été précédemment attribué à cette coordonnée. Les projets doivent être placés dans les limites du terrain. Le fichier de sortie contient au maximum N ordres. Si les instructions du fichier de sortie ne consomment pas tous les projets du tas, on suppose que tous les dossiers restants sont jetés (R).

Score

Toute solution non valide (voir remarques) ne donnera aucun points. Afin de calculer votre score, votre algorithme sera exécuté sur de nombreux différents fichiers d'entrée. Si le résultat est correct, des points seront attribués selon la longueur de la muraille. Si votre solution exécute une opération incorrecte, votre score sera nul. Dans tous les cas, votre algorithme sera arrêté après 2 secondes.

- 80% des points seront attribués sur des données contenant moins de 10 000 éléments à placer ;
- 20% des points seront attribués sur des données de plus de 10 000 éléments.