be-OI 2011 **Finale** 30 mars 2011	**Remplissez ce cadre en MAJUSCULES et LISIBLEMENT, svp** PRÉNOM NOM : ..... ÉCOLE : ..... SALLE : CANDIX / DAO MACHINE N°	**Réservé**

Olympiades belges d'Informatique (durée : 2h maximum)

Ce document est le questionnaire de **la partie machine** de la finale des Olympiades belges d'Informatique pour **la catégorie secondaire**. Il comporte trois questions. La première ne rapportera aucun point mais servira d'entrainement pendant le premier **quart d'heure**. Les deux suivantes ne seront distribuées qu'après ce quart d'heure et devront être résolues en **1h45 au maximum**. Seul le code soumis par le participant sur le site web de l'épreuve sera pris en compte pour l'évaluation de cette partie.

Délivrables

1. Votre programme lit les paramètres et données à traiter depuis l'entrée standard et doit écrire son résultat sur la sortie standard. Vous devez rendre votre programme via la système de soumission. Tous les détails concernant ces deux points 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.
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 **n'avez pas** accès à Internet durant l'épreuve. Toute tentative de communication avec d'autres participants ou toute autre personne extérieure sera sanctionnée.
6. Vous **pouvez** utiliser toutes les fonctionnalités de la librairie standard du langage que vous aurez choisi (parmi Java, C, C++, Pascal, Python et PHP) excepté tout ce qui implique une communication avec le monde extérieur hors entrée et sortie standards. En pratique, vous ne pouvez donc pas accéder au réseau, ni lire ou écrire des fichiers sur le disque.
7. Vous pouvez demander des **feuilles de brouillon** aux surveillants.
8. Il est strictement **interdit de manger ou boire** dans les salles informatiques. 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 deux heures** pour résoudre cet énoncé.

Bonne chance !

Questionnaire finale machine secondaire

Instructions pratiques

Ces instructions détaillent comment travailler sur votre programme et ensuite soumettre ce que vous avez écrit sur le serveur officiel. Nous vous conseillons de lire ceci tout en l'appliquant à la *Tâche 0*.

Étape 1 – Ouvrir le squelette du programme à compléter

- **Ouvrez votre dossier personnel** en double-cliquant sur l'icone *« Dossier personnel de olymp-sec »* sur le bureau.
- Celui-ci contient un répertoire (dossier) intitulé `OI2011`. **Ouvrez-le**.
- Le répertoire `OI2011` contient un répertoire par problème. **Ouvrez le répertoire de la question** sur laquelle vous voulez travailler.
- Dans le répertoire de chaque problème, vous trouverez un dossier par langage de programmation. **Ouvrez celui qui correspond au langage** dans lequel vous allez programmer (vous pourrez toujours changer par après).
- Dans ce répertoire, vous trouverez **un fichier source** nommé `code.*` (l'extension dépendant du langage de programmation), **un fichier d'entrée** pour votre programme (nommé `input-example.txt`) et **un fichier de sortie** correspondant à ce dernier (nommé `output-expected.txt`). Le fichier d'entrée contient un exemple d'entrée valide pour lequel votre programme devrait retourner le contenu du fichier de sortie fourni. Ces deux fichiers seront utilisés pour tester votre programme en conséquence.
- Double-cliquez sur le fichier source. Le programme *gedit* s'ouvre pour lire ce fichier.

Étape 2 – Compiler, exécuter et tester votre programme

- Lorsque vous êtes dans *gedit*, pressez **CTRL+R** afin de compiler, d'exécuter et tester votre programme.
- Une console s'affiche dans la partie inférieure de *gedit* et vous affiche, pour commencer, le déroulement de la compilation.
- Si votre programme s'est compilé avec succès, votre programme s'exécute avec comme entrée le fichier `input-example.txt` présent dans le dossier. Vous pouvez bien entendu modifier ce fichier ! Le résultat fournit par votre programme est alors écrit dans le fichier `output.txt`.
- Si l'exécution s'est déroulée sans erreur, la sortie de votre programme `output.txt` est comparée avec le résultat attendu se situant dans le fichier `output-expected.txt`. Bien entendu, si vous avez mis à jour `input-example.txt`, n'oubliez pas d'adapter `output-expected.txt`.
- Dans tous les cas, si une erreur survient (compilation ou exécution), des informations seront affichées dans le terminal de *gedit*.

Étape 3 – Soumettre votre programme

- **Ouvrez Firefox** (dans le menu *« Application > Internet > Firefox »*).
- Dans la barre d'adresse, entrez `http://130.104.78.201`.
- Connectez-vous sur ce site avec le login et mot de passe se trouvant sur la première page de ce questionnaire.
- Sélectionnez, sur la droite, la question que vous voulez soumettre.
- Cliquez sur *« submit a new solution »* pour soumettre un nouveau code. Si vous désirez repartir d'une précédente soumission, sélectionnez-la et cliquez sur *« See submitted source file »* pour l'éditer. Cela créera dans chacun des cas une nouvelle soumission.
- Choisissez votre langage de programmation (indispensable !). Ensuite, collez votre code dans le champ

principal ou modifiez celui existant. Enfin, appuyer sur *« submit »*.

- Votre soumission est alors mise en file d'attente. Cliquez sur *« see the result »* après quelques secondes pour voir le résultat. Si l'état est *« waiting »*, cela signifie que votre soumission est toujours dans la file. Si l'état est *« running »*, votre code est en cours d'exécution. Dans les deux cas, attendez quelques secondes (voire quelques minutes) avant de cliquer sur *« refresh »* afin de rafraichir la page. Il est inutile de rafraichir frénétiquement la page, cela ne peut que ralentir l'exécution du site pour tout le monde.
- Lorsque votre programme a été exécuté, vous trouverez sur cette même page le résultat de votre programme pour chacuns des tests. Pour chaque test, *« timeout »* signifie que votre programme a dépassé le temps d'exécution autorisé sur le serveur, *« error »* signifie qu'il y a une erreur de compilation, d'exécution ou que vous programme utilise des fonctions non-autorisées, *« wrong output »* signifie que la sortie de votre programme n'est pas celle attendue. Si votre programme dépasse le temps d'exécution pour un test (*« timeout »*), tous les tests suivants ne seront pas exécutés.

Fonctionnement du système de soumission

- Lorsque vous soumettez un nouveau code, celui-ci est mis dans une file d'attente et sera exécuté plus ou moins vite selon le nombre d'autres participants ayant soumis récemment. Il est donc possible qu'en fin de séance, le temps à attendre pour que votre programme soit exécuté atteigne plusieurs minutes.
- Si vous soumettez un nouveau code alors qu'une de vos soumissions précédentes à la même question est encore dans la file, cette dernière sera annulée et la nouvelle soumission remise en fin de file.
- Si à la fin des 2 heures, vous avez encore une soumission en attente, celle-ci sera bien entendu prise en compte. Par contre, vous ne saurez pas combien de points elle vous a rapportée.
- **Seul le meilleur score de chaque tâche sur toutes vos soumissions sera pris en compte pour calculer votre score final à la partie machine.**
- Avant de soumettre faites bien attention à retirer tout message imprimé à la console que vous auriez ajouté pour debugguer votre programme. Ceux-ci étant imprimés à la sortie standard, ils vont rendre le résultat de votre programme incorrect !

Remarques

- Les documentations pour chaque langage se trouvent dans le répertoire `OI2011/docs`.
- Si vous avez besoin d'aide concernant certains points de cette page, demandez de l'aide à un surveillant.
- Pour tout code non encore soumis sur la plateforme de soumission, il peut être utile d'en faire une copie dans un second fichier pour prévenir les erreurs de manipulation. Ces *« backups »* sont de votre ressort.
- Le code dans *gedit* ne sera jamais évalué, seul le code soumis sur le serveur a de la valeur pour votre score final.

Vous n'avez que deux heures pour cette épreuve. Préférez une solution peu performante qui fonctionne à une solution ambitieuse qui ne fonctionne finalement pas !

Tâche 0 – Sum

Voici un petit exercice qui vous permettra de tester et de prendre en main votre espace de travail, ainsi que le serveur de soumission. Il s'agit simplement de faire la somme de deux nombres entiers positifs.

Tâche

Écrire un programme qui, étant donné deux entiers positifs, calcule leurs somme.

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.

- $1 \leq a, b \leq 100$, les nombres à additionner ;

Quoi qu'il arrive, votre programme sera arrêté après **1 seconde** d'exécution. Votre programme ne peut utiliser plus de **10 Mo** de mémoire.

Entrée

L'entrée donnée à votre programme aura le format suivant :

- La première et unique ligne comporte deux entiers positifs a et b séparés par une espace unique ;
- L'entrée se termine par un saut de ligne.

Sortie

Votre programme écrit en sortie la somme de a et b, suivie d'un saut de ligne.

Exemple

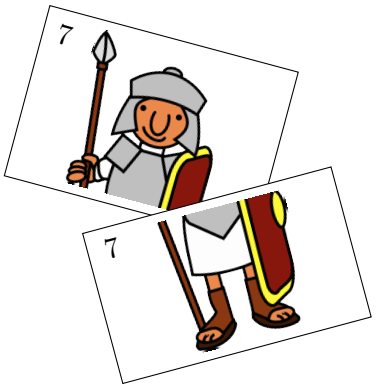
Soit l'entrée suivante fournie à votre programme :

```
10 81
```

La sortie de votre programme doit être :

```
91
```

Tâche 1 – Stickers (30 min)



Une toute nouvelle collection de 84 autocollants a été mise en place par PANINI© autour du thème d'*« Harry Potter et les Reliques de la Mort »*. Ces autocollants ont la particularité que l'image qu'ils contiennent ne représente pas un personnage complet, mais seulement une moitié de personnage. Quarante-deux autocollants représentent le haut des personnages et quarante-deux autres autocollants représentent le bas. Deux autocollants (haut et bas) qui correspondent au même personnage possèdent le même numéro.

Vous aimeriez savoir combien de personnages complets vous pouvez former avec votre collection d'autocollants. Pour ce faire, vous faites appel à votre frère Adrien pour vous aider. Vous prenez tous les autocollants correspondant à des parties hautes et votre frère prends ceux correspondant à des parties basses. Vous aimeriez maintenant faire le compte des personnages complets, le plus rapidement possible.

Tâche

Écrire un programme qui, étant donné deux listes d'entiers positifs L_1 et L_2 de longueurs M et N, représentant respectivement vos autocollants et ceux de votre frère Adrien, calcule le nombre maximal de personnages complets qu'il est possible d'avoir. Pour avoir le score maximum, votre programme doit être capable de passer tous les tests se situant dans les contraintes et limites précisées ci-dessous.

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.

- $1 \leq M, N \leq 200\,000$, pour la taille des jeux de cartes ;
- $0 \leq X < 42$, pour les numéros des cartes.

Quoi qu'il arrive, votre programme sera arrêté après **3 secondes** d'exécution. Votre programme ne peut utiliser plus de **256 Mo** de mémoire.

Entrée

L'entrée donnée à votre programme aura le format suivant :

- La première ligne comporte deux entiers positifs M et N : le nombre d'autocollants de L_1 et celui de L_2 ;
- Les M lignes suivantes contiennent chacune un entier positif, représentant les éléments de L_1 ;
- Les N dernières lignes contiennent chacune un entier positif, représentant les éléments de L_2 ;
- L'entrée se termine par un saut de ligne.

Sortie

Votre programme écrit en sortie le nombre maximal de personnages complets qu'il est possible de former avec les listes L_1 et L_2 d'autocollants, suivi d'un saut de ligne.

Exemple

Soit l'entrée suivante de votre programme, qui représente un jeu de 9 cartes et un second de 4 cartes :

```
9 4
7
11
1
7
2
3
9
3
9
8
7
2
7
```

La sortie de votre programme doit être :

```
3
```

Score

Le score total pour cette tâche est réparti sur deux jeux de données, chacun de ceux-ci comptant pour 50% des points total de la tâche.

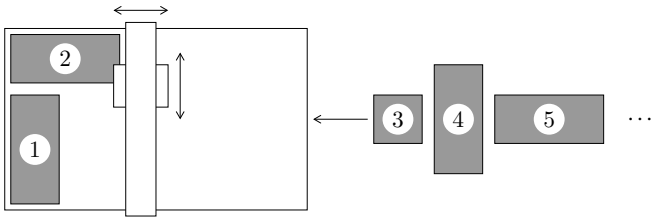
1. $0 \leq M, N \leq 200$ **(5 tests)**
2. $0 \leq M, N \leq 200\,000$ **(5 tests)**

Tâche 2 – Storage (60 min)

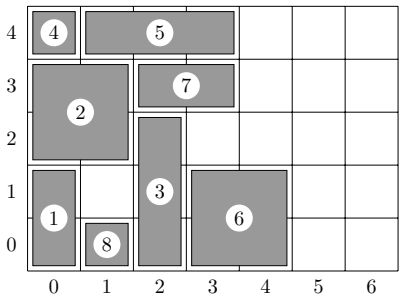
Vous êtes responsable d'un entrepôt situé dans le port d'Anvers. Votre boulot consiste à ranger des containers rectangulaires dans cet entrepôt, à l'aide d'un grue qui se déplace par dessus l'entrepôt. Afin d'optimiser le nombre de containers stockés dans l'entrepôt, vous appliquez la règle suivante :

- Les containers doivent toujours être placés le plus au fond possible de l'entrepôt (le plus à gauche) ;
- Lorsqu'il y a plusieurs possibilités le plus au fond, il faut prendre celle telle que le container se retrouve le plus au sud de l'entrepôt (le plus bas) ;

Bien entendu, les containers ne peuvent pas se superposer. De plus, ils ne peuvent pas être tournés et doivent être placés de manière à ce qu'ils soient parallèles aux murs de l'entrepôt.



Des camions arrivent à l'entrepôt et vous devez ranger les containers dans l'ordre dans lequel ils vous parviennent. Pour cette tâche, l'entrepôt est placé dans un repère cartésien, tel que la coordonnée du coin inférieur gauche soit $(0, 0)$ comme illustré sur la figure suivante. La position des containers sera identifiées par les coordonnées cartésiennes du coin inférieur gauche du container. Par exemple, le container 2 se trouve en $(0, 2)$ tandis que le container 7 se trouve en $(2, 3)$.



Tâche

Écrire un programme qui, étant donné un entrepôt de largeur L et de hauteur H (vue du dessus), et étant donné une liste de N containers de dimensions données, calcule la position du container dans l'entrepôt, sachant que ces derniers doivent toujours être placés le plus à gauche possible, et ensuite, le plus vers le bas (vue du dessus) possible.

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.

- $1 \leq L, H \leq 1\,000$, les dimensions de l'entrepôt ;
- $1 \leq N \leq 500$, le nombre de containers ;
- $1 \leq S_i, T_i \leq 50$, la taille du containers i ;

Quoi qu'il arrive, votre programme sera arrêté après **5 secondes** d'exécution. Votre programme ne peut utiliser plus de **256 Mo** de mémoire.

Entrée

L'entrée donnée à votre programme aura le format suivant :

- La première ligne comporte trois entiers positifs L, H et N : la longueur et largeur de l'entrepôt, ainsi que le nombre total de containers ;
- Les N lignes suivantes comporte chacune deux entiers positifs S et T, séparés par une espace unique : la largeur et la hauteur (vue du dessus) du container ;
- L'entrée se termine par un saut de ligne.

Sortie

La sortie à produire contient N lignes. Chacune de ces lignes contient deux entiers positifs qui représentent les coordonnées où placer un container, séparés par une espace unique. La coordonnée de la i^{e} ligne correspond au i^{e} container. La coordonnée correspond à la position du coin inférieur gauche du container. La sortie se termine par un saut de ligne.

Exemple

Soit l'entrée suivante qui représente un entrepôt de 7 de large sur 5 de hauteur (vue du haut) dans lequel il faudra ranger 8 containers dont les dimensions sont successivement 1×2, 2×2, $1 \times 3 \ldots$ Notez que toutes les entrées que l'on donnera à votre programme ont une solution.

```
7 5 8
1 2
2 2
1 3
1 1
3 1
2 2
2 1
1 1
```

Votre programme doit écrire en sortie (la solution est unique !) :

```
0 0
0 2
2 0
0 4
1 4
3 0
2 3
1 0
```

Score

Le score total pour cette tâche est réparti sur quatre jeux de données, chacun de ceux-ci comptant pour 25% des points total de la tâche.

1. Tous les containers sont de dimension 1×1. De plus, $N \leq 25, L \leq 25$ et $H \leq 25$ **(5 tests)**
2. Tous les containers ont les mêmes dimensions. De plus, $N \leq 25, L \leq 50$ et $H \leq 50$ **(5 tests)**
3. Les containers sont de dimensions quelconques. De plus, $N \leq 25, L \leq 50$ et $H \leq 50$ **(5 tests)**
4. Les containers sont de dimensions quelconques. De plus, $N \leq 500, L \leq 1000$ et $H \leq 1000$ **(5 tests)**

Pour chacun des jeux de données, 80% des points seront attribués linéairement en fonction du nombre de containers bien placés. Dès qu'un container est mal placé, les suivants ne sont plus comptabilisés. Les 20% de points restants seront attribués directement si tous les containers sont bien placés.