be-OI 2014 **Finale** 8 février 2014		

Olympiade belge d'Informatique (durée : 3h30 maximum)

Ce document est le questionnaire de **la partie machine** de la finale de l'Olympiade belge d'Informatique. Il comporte quatre questions. La première ne rapportera aucun point mais servira d'entrainement pendant le premier **quart d'heure**. Les trois suivantes ne seront distribuées qu'après ce quart d'heure et devront être résolues en **3h30 au maximum**.

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

1. 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 ou via la plateforme de soumission (voir page suivante). 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 au multi-threading et 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. Vous **ne pouvez pas quitter votre place** pendant l'épreuve. Si vous devez absolument vous rendre aux toilettes, faites-en la demande à un surveillant. Ce dernier pourra décider d'accepter ou de refuser votre requête selon qu'il soit possible, ou pas, de vous accompagner tout en assurant la surveillance de l'épreuve.

Délivrables et calcul des points

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 en ligne. Seul le code soumis sera utilisé pour calculer votre score final. Le code en tant que tel (sa qualité) ne sera pas utilisé par les évaluateurs.

Chaque tâche est séparée en sous-tâches chacune composée d'un ensemble de tests. Des algorithmes corrects mais non-optimaux peuvent ainsi marquer des scores partiels. Cependant vous ne marquerez les points indiqués pour une sous-tâche que si tous les tests de celle-ci passent. Dans le cas contraire, vous ne marquerez aucun point pour cette sous-tâche. Si une sous-tâche échoue, les sous-tâches suivantes sont tout de même évaluées. Les ensembles de tests sont construits de telle façon à ce qu'un même algorithme marque toujours le même nombre de points, quelque soit le langage utilisé.

Tout candidat finissant le concours (obtenant le score maximal) avant la fin de l'épreuve remportera **1 point supplémentaire par tranche de 10 minutes** restante avant la fin de l'épreuve. En cas d'égalité sur le score total de la finale, le score sans ces bonus sera pris en compte pour déterminer les positions finales. En cas de nouvelle égalité, l'heure de soumission du code vous ayant permis d'obtenir le score final sera utilisée.

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*.

Charger et modifier les squelettes de programme

- La page *énoncé* de chaque tâche contient un fichier zip qui contient le squelette des programmes pour chacun des langages de l'épreuve. Téléchargez ce fichier. Tous les fichiers téléchargés arrivent par défaut dans le répertoire *Téléchargements*. Cliquez sur l'icone de dossier dans la barre de gauche (Dossier personnel) dans lequel vous trouverez ce répertoire *Téléchargements*. Vous pouvez dépacer un fichier que vous avez téléchargé en le glissant, par exemple sur votre bureau.
- Faites un clic droit sur fichier zip puis *Extraire ici*. Dans le répertoire créé (par exemple `Q0`), vous trouverez un squelette de programme pour chaque langage de programmation (par exemple `task.c`), un exemple de données d'entrée (`input.txt`) et la sortie attendue pour cette entrée (`output-expected.txt`).
- Choisissez le langage dans lequel vous voulez travailler et double cliquez sur le fichier correspondant. Par défault, il s'ouvre dans l'éditeur de code *gedit*. D'autres éditeurs sont installés si vous préferez. Pour les ouvrir, tapez leur nom dans la barre de recherche disponible en cliquant sur la première icone de la barre de gauche. Nous vous recommandons de travailler sous gedit qui a été spécialement configuré pour l'épreuve.
- Le squelette fourni contient le code permettant de lire les données d'entrée du problème et d'écrire le résultat en sortie. Vous devez compléter ce squelettre avec l'algorithme qui résout le problème.

Compiler, exécuter et tester votre programme

- Lorsque vous êtes dans *gedit*, pressez **Ctrl+R** pour tester le programme que vous éditez. Cette commande sauve sur disque, compile, exécute et teste votre programme avec le fichier `input.txt` comme données. Les résultats s'affichent dans une console dans la partie inférieure de *gedit*.
- Si votre programme s'est compilé avec succès, votre programme s'exécute avec comme entrée le fichier `input.txt` présent dans le dossier. Si l'exécution s'est déroulée sans erreur, la sortie de votre programme se trouve dans le fichier `output.txt` et est comparée avec le résultat attendu se situant dans le fichier `output-expected.txt`. Dans tous les cas, si une erreur survient (compilation ou exécution), des informations seront affichées dans la console de *gedit*.
- Vous pouvez modifier `input.txt` pour effectuer d'autres tests, mais dans ce cas vous devez bien entendu adapter le fichier de sortie attendu `output-expected.txt`.

Soumettre votre programme

- La soumission se fait en envoyant votre fichier sur le serveur. Sur le serveur, rendez-vous sur la page *Soumissions* de la tâche correspondante. Cliquez sur *Parcourir* et sélectionnez le fichier dans lequel vous avez travaillé et ensuite *Soumettre*.
- Les résultats de votre soumission apparaissent sur cette même page dans la colonne *Score public*. Votre score final pour chacune des tâches sera le score maximal que vous avez obtenu pour cette question.
- Vous pouvez obtenir un détail de votre score par sous-tâche en cliquant sur *Détails* ainsi que la durée d'exécution pour chacun des tests.
- Seul le résultat de votre programme aux tests automatiques déterminera votre résultat, la qualité de votre code n'entre donc absolument pas en ligne de compte.

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.
- Vous ne pouvez pas soumettre un code si vous en avez déjà soumis un il y a moins de **20 secondes**. Attention aux dernières minutes de l'épreuve !
- Si à la fin du temps de l'épreuve 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 rapportés.
- **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 sur la plateforme (web) de soumission, dans la section *Documentation*.
- Vous pouvez poser une question aux organisateurs du concours via la page *Communication* du serveur de soumission. Vous serez automatiquement notifié lorsqu'une réponse vous parviendra.
- 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.
- **Problème connu :** Après avoir lancé le test de votre programme avec **Ctrl+R**, il peut arriver que le curseur de la souris se comporte comme si le test était toujours en cours. Il s'agit d'un bug de gedit. Le test est néanmoins bien terminé et vous pouvez continuer à travailler sereinement. Le curseur de gedit revient dans son état normal après un certain délai.
- Si un message concernant l'intégration de la souris est affiché en haut de votre écran, vous pouvez l'ignorer (le fermer), il s'agit d'un message lié à l'environnement virtuel dans lequel vous travaillez.
- Vous aurez peut-être remarqué une page *Tests* sur le serveur de soumission. Celle-ci vous permet de soumettre votre programme et un fichier d'entrée personnel pour qu'il soit évalué sur le même serveur que celui d'évaluation. Vous pouvez utiliser cette fonctionnalité si vous le souhaitez.

Tâche 0 – (0 pts)

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 leur somme.

Limites et contraintes

- $1 \leq a, b \leq X_{MAX}$, les nombres à additionner ;

	X_{MAX}
Sous-tâche A	100
Sous-tâche B	2 000 000

Durée maximale d'exécution : **1 seconde**. Limite mémoire : **256 Mo**.

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 un 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 – Vagues de chaleur (Heat) – (30 pts) (A : 11 pts, B : 19 pts)

Les informaticiens sont connus pour préférer rester à l'intérieur face à leur ordinateur plutôt que de profiter du soleil lors des chauds mois d'été. Une nouvelle étude en a découvert la principale raison : les informaticiens supportent mal la chaleur et peuvent perdre toute leur capacité de réflexion s'ils sont exposés plusieurs jours de suite à une température supérieure à un certain seuil propre à chacun.

Le comité de l'Olympiade Internationale d'Informatique, alerté par cette découverte, veut prendre des mesures afin d'améliorer la sélection des pays accueillant l'épreuve. Il vous demande donc d'écrire un programme capable de mesurer l'indice d'inconfort d'un participant pour une destination. Cet indice est défini comme le nombre maximal de jours consécutifs durant lesquels la température est supérieure ou égale à la température seuil du participant. Vous vous êtes donc procuré les températures maximales journalières mesurées ces derniers étés pour toutes les prochaines villes candidates à l'organisation de l'IOI, ainsi que le seuil de température supporté par chacun des futurs candidats.

Tâche

Vous devez écrire l'algorithme qui permet, à partir du seuil d'inconfort d'un participant et d'une suite de températures, de calculer la longueur de la plus longue suite contiguë de températures supérieures ou égales à ce seuil.

Limites et contraintes

- $0 < N < N_{MAX}$, le nombre de jours auxquels la température a été mesurée.
- $0 < H < T_{MAX}$, le seuil du participant. Ce dernier ne supportera pas les températures supérieures ou égales à ce seuil.
- $0 < T_i < T_{MAX}$, la température maximale au jour i

	N_{MAX}	T_{MAX}
Sous-tâche A	50	10 000
Sous-tâche B	100 000	10 000

Durée maximale d'exécution : **2 secondes**. Limite mémoire : **128 Mo**.

Entrée

L'entrée donnée à votre programme aura le format suivant :
- Une première ligne contenant N et H, séparés par un espace.
- N lignes contenant T_i
- L'entrée se termine par un saut de ligne.

Sortie

La sortie de votre programme contient un seul nombre, la longueur de la plus longue vague de chaleur. Le fichier se termine par un saut de ligne.

Exemple 1

Pour l'entrée suivante

```
13 38
31
39
42
33
30
33
40
38
39
41
37
34
27
```

Votre programme renverra :

```
4
```

Exemple 2

Pour l'entrée suivante

```
5 40
28
39
38
27
34
```

Votre programme renverra :

```
0
```

Tâche 2 – Feu d'artifice (Firework) – (60 pts) (A : 23 pts, B : 19 pts, C : 18 pts)

A Taïwan, comme dans toute cette partie de l'Asie, il est très courant de tirer des feux d'artifice pour fêter toute sorte d'évènement. A l'occasion du lancement de l'Olympiade internationale, les organisateurs aimeraient programmer un grand feu d'artifice et ont besoin de vous pour obtenir le meilleur résultat possible.

Vous avez un certain nombre de fusées à votre disposition. Ces fusées vous sont données dans un certain ordre fixé. Vous pouvez commencer le spectacle par la fusée que vous souhaitez et utiliser autant de fusées que vous le souhaitez mais vous ne pouvez pas changer l'ordre dans lesquelles elles sont lancées : une fois qu'une fusée a été tirée, celles qui la précèdent ne peuvent plus être tirées. Chaque fusée a également les propriétés suivantes :

— x : la position de la fusée le long de l'horizon
— $[a, b]$: l'intervalle d'harmonie de la fusée (expliqué ci-dessous)
— p : l'indice de beauté de la fusée

Afin d'obtenir le meilleur feu d'artifice, vous devez vous assurer que chaque fusée que vous utilisez a un intervalle d'harmonie contenant la position, le long de l'horizon, de la fusée tirée juste avant. Par exemple, si vous voulez tirer une fusée avec l'intervalle d'harmonie $[a_j, b_j]$, vous devez vous assurer que la fusée précédente a une position x_{j-1} comprise entre a_j et b_j (inclus). Bien entendu, vous pouvez tirer n'importe quelle fusée comme première fusée vu que celle-ci n'est précédée par aucune autre. La beauté totale du spectacle est la somme des indices de beauté des fusées tirées.

Tâche

Vous devez déterminer le score de beauté maximal qu'il est possible d'atteindre étant donné une configuration de fusées.

Limites et contraintes

- $1 \leq N \leq N_{MAX}$, le nombre de fusées.
- $0 \leq a_i \leq b_i \leq 500\,000$, l'intervalle d'harmonie de la fusée i
- $0 \leq x_i \leq 500\,000$, la position le long de l'horizon de la fusée i
- $0 < p_i < P_{MAX}$, l'indice de beauté de la fusée i.

	N_{MAX}	P_{MAX}
Sous-tâche A	10	100 000
Sous-tâche B	1 500	100 000
Sous-tâche C	25 000	1 000 000

De plus pour la sous-tâche C, la longueur des intervalles d'harmonie est toujours inférieur à 20 et la solution est inférieure à 2 milliards.

Durée maximale d'exécution : **2,5 secondes**. Limite mémoire : **512 Mo**.

Entrée

L'entrée donnée à votre programme aura le format suivant :
- Une première ligne contenant N.
- N lignes contenant chacune x_i a_i b_i p_i (séparés par un espace à chaque fois) décrivant les propriétés de la fusée i.
- L'entrée se termine par un saut de ligne.

Sortie

Le score maximal de beauté qu'il est possible d'obtenir. Le fichier se termine par un saut de ligne.

Exemple 1

Pour l'entrée suivante

```
5
9 0 6 4
5 7 8 1
7 0 6 2
2 1 5 3
9 3 8 2
```

Votre programme renverra :

```
5
```

Explication : La fusée 1 est tirée à la position 9. Aucune fusée après celle-ci n'inclut cette position dans son intervalle d'harmonie. Le meilleur feu d'artifice qu'il est donc possible de composer en commençant par cette fusée est donc de 4. La fusée 2 peut, elle, être suivie par la fusée 3, 4 ou 5 car $[0, 6]$, $[1, 5]$ et $[3, 8]$ couvre sa position (5). De plus, la fusée 5 peut également être tirée après la fusée 3. La séquence 2, 3, 5 donne un total de beauté de $1 + 2 + 2 = 5$, qui est le maximum possible.

Exemple 2

Pour l'entrée suivante

```
6
75 10 99 100
15 12 72 100
30 24 72 100
45 11 14 100
60 31 44 100
72 62 72 100
```

Votre programme renverra :

```
100
```

Explication : Quelque soit la première fusée tirée, il n'y a pas de fusée parmi les suivantes contenant celle-ci dans son intervalle d'harmonie.

Tâche 3 – La rizière de Mr. Chen (Paddy) – (60 pts) (A : 20 pts, B : 20 pts, C : 20 pts)

M. Chen est un grand propriétaire terrien de la conté du Taoyuan, au sud-ouest de Taipei. Il désire créer une nouvelle rizière, la plus grande possible, au milieu de son terrain. Cependant, sa rizière doit impérativement être rectangulaire et entourée par une barrière afin d'éviter les intrusions.

La seule barrière dont dispose M. Chen est longue barrière d'un seul tenant, et composée de sections de longueurs diverses, reliées les unes aux autres par des articulations. M. Chen aimerait optimiser la superficie de son champ et a besoin pour cela de votre aide.

Tâche

Etant donnés les longueurs des sections de la barrière, calculer la surface maximale que la rizière rectangulaire pourra faire en utilisant l'ensemble des sections de la barrière et sans changer leur ordre. S'il n'est pas possible de former un champ rectangulaire, votre programme doit retourner 0.

Indice : La forme de rectangle maximisant sa surface pour un même périmètre sera toujours celle qui se rapproche le plus d'un carré.

Limites et contraintes

- $1 \leq N \leq N_{MAX}$, le nombre de sections de la barrière ;
- $1 \leq L_i \leq 1\,000$, la longueur de la section i de la barrière ;
- $1 \leq T \leq T_{MAX}$, la longueur totale de la barrière ;

	N_{MAX}	T_{MAX}
Sous-tâche A	30	30 000
Sous-tâche B	1 000	100 000
Sous-tâche C	100 000	100 000 000

Notez que pour la sous-tâche C, la surface totale pourra faire jusqu'à $6,25 \times 10^{14}$, et qu'il est donc nécessaire de stocker cette surface dans une variable suffisamment grande (dépendant de votre langage de programmation).

Durée maximale d'exécution : **4 secondes**. Limite mémoire : **512 Mo**.

Entrée

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

- La première ligne comporte un entier positif N ;
- La N lignes suivantes comportent chacune un nombre entier L_i ;
- L'entrée se termine par un saut de ligne.

Sortie

Votre programme écrit en sortie la surface maximale de rizière rectangulaire qu'il est possible de créer avec la barrière (d'un seul tenant) fournie. S'il n'est pas possible de créer un enclos rectangulaire, votre programme retourne 0. Le fichier se termine par un saut de ligne.

Exemple 1

Soit l'entrée suivante fournie à votre programme :

```
7
2
4
1
2
3
1
1
```

La sortie de votre programme doit être :

```
12
```

Explication : Le plus grand enclos possible est un enclos de taille 3 sur 4. Le premier côté est formé d'une section de longueur 4 (L_2), le second des sections 1 (L_3), 2 (L_4), le troisième des sections 3 (L_5), 1 (L_6) et le dernier des sections 1 (L_7) et 2 (L_1). Il était également possible de créer un enclos de taille 6 sur 1, mais sa superficie était plus petite.

Exemple 2

Soit l'entrée suivante fournie à votre programme :

```
5
2
3
2
3
2
```

La sortie de votre programme doit être :

```
0
```

Explication : Il n'est pas possible de former un rectangle en utilisant toutes les sections de la cloture.