be-OI 2012 **Finale** 14 mars 2012	**Remplissez ce cadre en MAJUSCULES et LISIBLEMENT, svp** PRÉNOM NOM : ..... ÉCOLE : ..... LOGIN : `INSERT-LOGIN` MOT DE PASSE : `INSERT-PASSWORD` SALLE : Babbage / Boole	**Réservé** SID

Olympiade belge d'Informatique (durée : 1h20 maximum)

Ce document est le questionnaire de **la partie machine** de la finale de l'Olympiade belge d'Informatique. 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 **1h20 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 et école sur la première page. Entourez le nom de la salle dans laquelle vous vous trouvez. 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 leurs places** pendant l'épreuve, par exemple pour aller aux toilettes ou pour fumer une cigarette.
9. Vous avez **exactement une heure vingt** pour résoudre cet énoncé.

Bonne chance !

Questionnaire finale machine

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*.

Ouvrir le squelette du programme à compléter

- **Ouvrez votre dossier personnel** en double-cliquant sur l'icone `be-oi12_final` sur le bureau.
- Ce répertoire `be-oi12_final` contient un répertoire par tâche (`task_0` à `task_2`) et un répertoire `docs`.
- Le répertoire `docs` contient un répertoire par langage où vous trouverez la documentation de ce langage.
- Dans le répertoire de chaque tâche, vous trouverez un dossier par langage de programmation. **Ouvrez celui qui correspond au langage** dans lequel vous désirez 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 la solution attendue à la tâche (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.
- Du code est déjà présent dans votre fichier afin de vous aider avec les entrées. Vous êtes libres de le modifier si vous le désirez.

Compiler, exécuter et tester votre programme

- Dans chaque répertoire, vous avez un exécutable nommé `oi-build.sh`. Vous pouvez double-cliquer sur celui-ci et choisir l'option `run in terminal`, une console (aussi nommé ligne de commande) apparaîtra avec 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-example.txt`.
- Si l'exécution s'est déroulée sans erreur, la sortie de votre programme `output-example.txt` est comparée avec le résultat attendu se situant dans le fichier `output-expected.txt`. Bien entendu, si vous avez mis à jour le fichier `input-example.txt`, n'oubliez pas d'adapter le fichier de sortie attendu `output-expected.txt`.
- Si votre programme produit le résultat attendu, un fichier nommé `tosubmit.zip` sera créé dans le répertoire où vous travaillez. Ce fichier zippé vous sera utile pour la soumission de votre programme sur le site de tests automatiques.
- Dans tous les cas, si une erreur survient (compilation ou exécution), des informations seront affichées dans le terminal.
- Il suffit de presser la touche `enter` pour quitter la ligne de commande.

Soumettre votre programme

- Dans le répertoire `be-oi12_final`, vous avez un raccourci vers le site de soumission, nommé `automate`.
- Connectez-vous sur ce site avec le login et mot de passe se trouvant sur la première page de ce questionnaire (le même que celui utilisé pour vous connecter à vos terminaux).
- Sélectionnez, sur la gauche, l'option **New Submission**.
- Vous pouvez choisir la question pour laquelle vous voulez soumettre en la sélectionnant dans la liste.

- Cliquez sur le bouton **browse** pour choisir le fichier zippé (créé lors de l'opération précédente) à soumettre.
- Cliquez sur le bouton **OK** pour soumettre votre solution.
- Pour connaître le résultat de vos soumissions, vous pouvez choisir l'option **Show exercise** sur la gauche.
- Pour chaque tâche, vous avez un résumé de son état, une description et si cette tâche est réussie (colonne *passed*). Vous pouvez cliquer sur la soumission (colonne *as part of submission* afin de voir les détails.
- Sur cette nouvelle page, le résultat aux tests est affiché en haut de la page (*ok* (réussi), *fail* (raté) et *in evaluation* (en attente)). D'autres informations intéressantes s'y trouvent, notamment :

  Data valid le fichier soumis a le bon format

  Failed to prepare evaluation le fichier n'a pas pu être compilé

  Failed to perform evaluation l'exécution du programme n'a pas pu se terminer normalement

  Evaluation done l'évaluation a pu se terminer
- Vous y trouvez également le détail des résultats de votre soumission en cliquant sur le lien *show* sous la colonne *détails*. Dans le cas d'un succès, vous ne verrez que la description de la tâche. Dans le cas d'un échec, vous y verrez le nombre de tests réussis.
- 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.
- 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 du temps imparti, 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 `be-oi12_FINAL/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 qu'une heure et vingt minutes 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 leur 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 – Synchronisation

Sur Internet, il est très courant de devoir synchroniser plusieurs serveurs entre eux, principalement entre un serveur principal et des serveurs miroirs. L'une des procédures communes consiste, pour un serveur miroir, à recevoir la liste des objets présents sur le serveur primaire et de calculer les objets qu'ils ne possèdent pas localement afin d'initier la copie de ceux-ci.

Nous vous demandons d'implémenter un algorithme permettant de détecter les éléments manquants entre deux listes, chaque élément étant identifié par un nombre entier unique.

Tâche

Écrire un programme qui, étant donné deux listes ordonnées d'entiers positifs A et B de longueurs N et M, retourne la liste des valeurs présentes dans A mais pas dans B. A contient au moins toutes les valeurs contenues dans B. Les deux listes ne contiennent pas de doublons. 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.

- $1 \leq N, M \leq 100\,000$, la taille des deux listes d'entiers ;
- $0 \leq A_i, B_i < 1\,000\,000$, les entiers compris dans ces listes.

Quoi qu'il arrive, votre programme sera arrêté après **1 minute** d'exécution. Votre programme ne peut utiliser plus de **128 Mo** de mémoire.

Entrée

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

- La première ligne comporte deux entiers positifs N et M : la taille de, respectivement, A et B ;
- Les M lignes suivantes contiennent chacune un entier positif différent, représentant les éléments de B triés par ordre croissant ;
- Les N dernières lignes contiennent chacune un entier positif différent, représentant les éléments de A triés par ordre croissant ;
- L'entrée se termine par un saut de ligne.

Sortie

Votre programme écrit en sortie la liste triée des entiers se trouvant dans A et non-présents dans B. Le fichier doit se terminer par un saut de ligne.

Exemple

Soit l'entrée suivante de votre programme :

```
6 2
4
7
1
4
6
7
10
12
```

La sortie de votre programme doit être :

```
1
6
10
12
```

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.

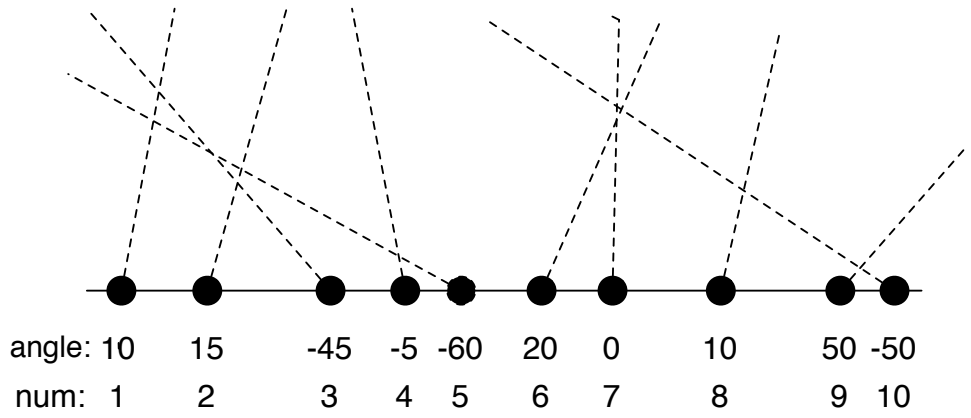
1. $0 \leq N, M \leq 200$ **(5 tests)**
2. $0 \leq N, M \leq 100\,000$ **(5 tests)**

Tâche 2 – Ghostbusters

Comme dans le célèbre film « SOS fantômes », une armée de revenants, esprits, et autres incubes maléfiques, s'apprête à envahir la ville de New York. Une équipe de chasseurs de fantômes (qui compte maintenant 10 membres) a été appelée à la rescousse.

Comme chacun le sait, les chasseurs de fantômes utilisent une arme spéciale, appelée *proton pack* qui émet un rayon de protons permettant de capturer le revenant recalcitrant. Afin de combattre efficacement les fantômes, les chasseurs se sont placés le long d'une ligne, pour faire face à leurs ennemis. Malheureusement, un démon plus malin que les autres a pris possession de l'esprit des chasseurs de fantômes, qui sont maintenant figés dans une position déterminée et ne peuvent plus décider dans quelle direction ils tirent. Par contre, ils ont encore la faculté de décider s'ils tirent ou non.

La situation (vue du dessus) se présente donc comme sur la figure ci-dessous. Chaque point sur la droite en trait continu représente un chasseur de fantômes (identifiés par les lettres A à J), et chaque trait en pointillé émanant d'un chasseur de fantôme représente le début du rayon de protons qu'il émettrait s'il se décidait à tirer. L'angle exact de tir est indiqué sous le point (il vaut « 0 » si le chasseur de fantôme tire devant lui, il est négatif s'il tire vers la gauche, positif s'il tire vers la droite).



Malheureusement, le danger des *proton packs* est qu'on ne peut sous aucun prétexte croiser les rayons émis par deux chasseurs de fantômes. Cela aurait des conséquences tellement terribles que nous n'osons même pas les détailler ici. Comme les chasseurs de fantômes ne peuvent pas contrôler dans quelle direction ils tirent, la seule solution pour éviter une telle catastrophe est d'en empêcher certains de tirer, de façon à garantir qu'aucun rayon n'en croise un autre. Naturellement, le nombre de chasseurs de fantômes qui tirent doit être le plus grand possible, afin d'être certain d'exterminer les revenants…

Tâche

Afin de pouvoir faire face aux prochaines attaques de fantômes (sait-on jamais ?), on vous demande de donner un algorithme qui donne, pour un nombre quelconque de chasseurs de fantômes, un sous-ensemble maximal de chasseurs de fantômes qui sont autorisés à tirer.

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.

- Votre algorithme recevra en entrée un ensemble de listes d'angles de tirs, exprimés en degrés, entiers, et compris entre -90 et 90 (bornes incluses).

- La liste comportera au maximum 10 000 entrées.
- Il y aura au maximum 10 000 listes

Entrée L'entrée donnée à votre programme aura le format suivant :
- Une première ligne indiquant le nombre ℓ d'instance à résoudre (c'est-à-dire le nombre de listes d'angles), puis un espace, puis le nombre k de chasseurs de fantômes.
- ℓ lignes contenant chacune k valeurs séparées par des espaces. La première valeur de la ligne représente l'angle de tir du chasseur numéro 1, la seconde, celui du chasseur numéro 2, etc.
- L'entrée se termine par un saut de ligne.

Sortie La sortie de votre programme aura le format suivant :
- ℓ lignes comprenant chacune au plus k valeurs, et représentant les numéros des chasseurs de fantômes devant tirer, triés par ordre croissant. Les chasseurs de fantômes sont numérotés de 1 à k.
- le fichier se termine par un saut de ligne.

Exemple

Pour l'entrée suivante, comprenant 2 séquences d'angles pour 6 chasseurs de fantômes :

```
2 6
65 43 6 31 57 -30
-49 44 7 -72 51 54
```

Votre programme renverra :

```
3 4 5
1 3 5 6
```