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 supérieur**. 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 supérieur

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-sup »* 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

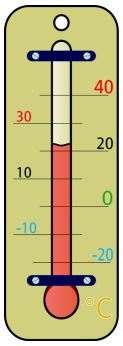
Soit l'entrée suivante fournie à votre programme :

```
10 81
```

La sortie de votre programme doit être :

```
91
```

Tâche 1 – Temperature

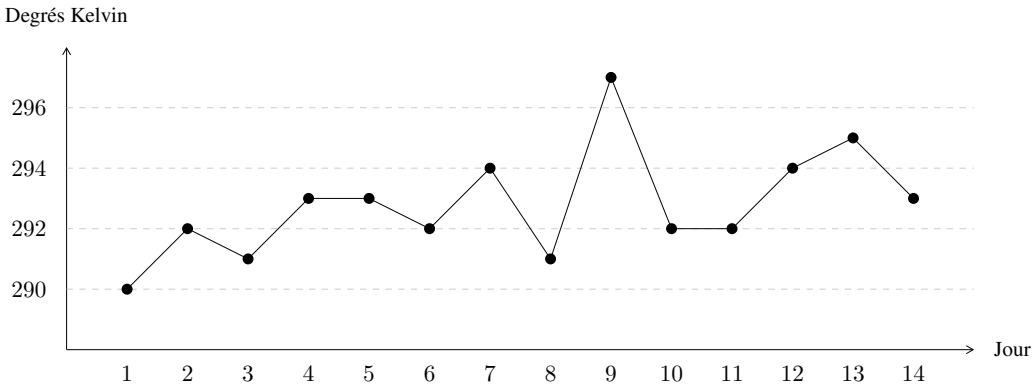


Vous avez été engagé comme analyste à l'Observatoire Royal de Belgique. Ce dernier a reçu une demande de la part de la Direction générale Statistique et Information économique qui aimerait identifier des périodes de temps durant lesquelles la température moyenne est la plus élevée.

À cette fin, vous avez reçu une série de données correspondant à des mesures de température prises tous les jours pendant une certaine période. On vous demande ensuite de chercher plusieurs choses :

- La température la plus élevée sur la période ;
- La température moyenne la plus élevée sur 7 jours successifs ;
- La température moyenne le plus élevée sur 31 jours successifs.

Afin de résoudre ce problème, vous avez pensé à un algorithme beaucoup plus général. Votre algorithme va prendre en paramètre le nombre de jours successifs dont vous voulez chercher la température moyenne maximale. Ce nombre de jour est appelé la *fenêtre d'observation*. Voici un exemple de données de températures (qui sont données en degrés Kelvin) :



Si on prend une fenêtre de 1, la moyenne maximale se retrouve le jour 9 et vaut **297**. Si on prend une fenêtre de 3, on se retrouve avec une moyenne maximale du jour 12 à 14 : $(294 + 295 + 293)/3 = \mathbf{294}$. C'est cette valeur que votre programme doit retrouver.

Tâche

Écrire un programme qui, étant donnés un entier positif F et un tableau tab de N entiers positifs, retourne la moyenne maximale qu'il est possible d'avoir parmi tous les sous-tableau de taille F qu'il est possible de former dans le tableau tab de données.

Limites

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

- $2 \leq F \leq 10\,000$;
- $F \leq N \leq 500\,000$;
- $0 \leq tab_i \leq 320$, les valeurs du tableau.

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 séparés par une espace unique : N et F ;
- Les N lignes suivantes consistent chacune en un entier positif : une mesure de température ;
- L'entrée se termine par un saut de ligne.

Sortie

La sortie à produire contient une seule ligne avec un entier positif qui est la moyenne maximale qu'il est possible d'obtenir avec une fenêtre de taille F (arrondie vers le bas). Ce nombre est suivi d'un saut de ligne.

Exemple

Soit l'entrée suivante qui représente 14 mesures de température et propose une fenêtre de 3 :

```
14 3
290
292
291
293
293
292
294
291
297
292
292
294
295
293
```

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

```
294
```

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.

- $2 \leq F \leq 25$ et $F \leq N \leq 1\,000$ **(5 tests)**
- $2 \leq F \leq 10\,000$ et $F \leq N \leq 500\,000$ **(5 tests)**

Chaque test n'a qu'une seule solution. Toute réponse correcte vous rapportera donc des points, alors que toute réponse incorrecte ne vous rapportera rien.

Tâche 2 – Repetition

Tâche

Écrire un programme qui, étant donnée une chaine de caractères de longueur N, retourne la chaine la plus longue répétée dans cette chaine de caractères. Si il y a plusieurs chaines de longueur maximale, la première est retournée. La chaine est composée de caractères majuscules et minuscules, de signes de ponctuation et d'espaces. Les majuscules et minuscules sont considérés comme différents.

Les deux chaines qui se répètent peuvent se superposer, par exemple, la plus longue sous-chaine répétée dans "lalala" est "lala".

Par exemple, si la chaine d'entrée est "Lorem ipsum dolor a porttitor lorem ac ipsum tempor dolor", le programme doit retourner " ipsum ". Notez les 2 espaces, qui sont également répétés.

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.

- $0 < N \leq 65\,530$
- Les caractères de la chaine comprennent les 26 lettres de l'alphabet en minuscule et majuscule, des espaces, des virgules, des points-virgules et des points.

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 contient un entier positif N représentant le nombre de caractères de la chaine à analyser ;
- La seconde ligne contient la chaine de caractères à analyser ;
- L'entrée se termine par un saut de ligne.

Sortie

La sortie à produire contient une seule ligne, la plus longue sous-chaine de caractères répétée. Cette sous-chaine se termine par un saut de ligne.

Exemple

Soit l'entrée suivante contenant une chaine de 124 caractères (notez que la chaine de caractères est bien sur une unique ligne (la seconde)) :

```
124
Lorem ipsum dolor sit amet, consectetur adipisicing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua.
```

Votre programme doit écrire en sortie (la solution est unique !) (notez l'espace en début de chaine) :

```
 dolor
```

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. $0 < N \leq 1\,000$ **(5 tests)**
2. $20\,000 < N \leq 30\,000$ **(5 tests)**
3. $45\,000 < N \leq 50\,000$ **(5 tests)**
4. $60\,000 < N \leq 65\,530$ **(5 tests)**

Chaque test n'a qu'une seule solution. Toute réponse correcte vous rapportera donc des points, alors que toute réponse incorrecte ne vous rapportera rien.