be-OI 2023 **Finale - SENIOR** samedi 18 mars 2023	**Remplissez ce cadre en MAJUSCULES et LISIBLEMENT svp** PRÉNOM : ..... NOM : ..... ÉCOLE :	O **Réservé**

Finale de l'Olympiade belge d'Informatique 2023 (durée : **2h au maximum**)

Notes générales (à lire attentivement avant de répondre aux questions)

1. Vérifiez que vous avez bien reçu la bonne série de questions (mentionnée ci-dessus dans l'en-tête):
   - Pour les élèves jusqu'en deuxième année du secondaire: catégorie **cadet**.
   - Pour les élèves en troisième ou quatrième année du secondaire: catégorie **junior**.
   - Pour les élèves de cinquième année du secondaire et plus: catégorie **senior**.
2. N'indiquez votre nom, prénom et école **que sur cette page**.
3. Indiquez **vos réponses** sur les pages prévues à cet effet. Écrivez de façon **bien lisible** à l'aide d'un **bic ou stylo** bleu ou noir.
4. Utilisez un crayon et une gomme lorsque vous travaillez au brouillon sur les feuilles de questions.
5. Vous ne pouvez avoir que de quoi écrire avec vous; les calculatrices, GSM, smartphone, ... sont **interdits**.
6. Vous pouvez toujours demander des feuilles de brouillon supplémentaires à un surveillant.
7. Quand vous avez terminé, **remettez la première page (avec votre nom) et les pages avec les réponses**, vous pouvez conserver les autres pages.
8. Tous les extraits de code de l'énoncé sont en **pseudo-code**. Vous trouverez, sur les pages suivantes, une **description** du pseudo-code que nous utilisons.
9. Si vous devez répondre en code, vous devez utiliser le **pseudo-code** ou un **langage de programmation courant** (Java, C, C++, Pascal, Python, ...). Les erreurs de syntaxe ne sont pas prises en compte pour l'évaluation.

Bonne chance !

L'Olympiade Belge d'Informatique est possible grâce au soutien de nos membres:

Aide-mémoire pseudo-code

Les données sont stockées dans des variables. On change la valeur d'une variable à l'aide de $\leftarrow$. Dans une variable, nous pouvons stocker des nombres entiers, des nombres réels, ou des tableaux (voir plus loin), ainsi que des valeurs booléennes (logiques): vrai/juste (**`true`**) ou faux/erroné (**`false`**). Il est possible d'effectuer des opérations arithmétiques sur des variables. En plus des quatre opérateurs classiques ($+$, $-$, $\times$ et $/$), vous pouvez également utiliser l'opérateur %. Si a et b sont des nombres entiers, alors a/b et $a\%b$ désignent respectivement le quotient et le reste de la division entière.
Par exemple, si $a = 14$ et $b = 3$, alors $a/b = 4$ et $a\%b = 2$.

Voici un premier exemple de code, dans lequel la variable age reçoit 17.

```
anneeNaissance ← 2006
age ← 2023 − anneeNaissance
```

Pour exécuter du code uniquement si une certaine condition est vraie, on utilise l'instruction **`if`** et éventuellement l'instruction **`else`** pour exécuter un autre code si la condition est fausse. L'exemple suivant vérifie si une personne est majeure et stocke le prix de son ticket de cinéma dans la variable $prix$. Observez les commentaires dans le code.

```
if (age ≥ 18)
{
    prix ← 8 // Ceci est un commentaire.
}
else
{
    prix ← 6 // moins cher !
}
```

Parfois, quand une condition est fausse, on doit en vérifier une autre. Pour cela on peut utiliser **`else if`**, qui revient à exécuter un autre **`if`** à l'intérieur du **`else`** du premier **`if`**. Dans l'exemple suivant, il y a 3 catégories d'âge qui correspondent à 3 prix différents pour le ticket de cinéma.

```
if (age ≥ 18)
{
    prix ← 8 // Prix pour une personne majeure.
}
else if (age ≥ 6)
{
    prix ← 6 // Prix pour un enfant de 6 ans ou plus.
}
else
{
    prix ← 0 // Gratuit pour un enfant de moins de 6 ans.
}
```

Pour manipuler plusieurs éléments avec une seule variable, on utilise un tableau. Les éléments individuels d'un tableau sont indiqués par un index (que l'on écrit entre crochets après le nom du tableau). Le premier élément d'un tableau $tab[\,]$ est d'indice 0 et est noté $tab[0]$. Le second est celui d'indice 1 et le dernier est celui d'indice $n-1$ si le tableau contient n éléments. Par exemple, si le tableau $tab[\,]$ contient les 3 nombres 5, 9 et 12 (dans cet ordre), alors $tab[0] = 5$, $tab[1] = 9$, $tab[2] = 12$. Le tableau est de taille 3, mais l'indice le plus élevé est 2.

Pour répéter du code, par exemple pour parcourir les éléments d'un tableau, on peut utiliser une boucle **for**. La notation **for** ($i \leftarrow a$ **to** b **step** k) représente une boucle qui sera répétée tant que $i \leq b$, dans laquelle i commence à la valeur a, et est augmenté de k à la fin de chaque étape. L'exemple suivant calcule la somme des éléments du tableau $tab[\,]$ en supposant que sa taille vaut n. La somme se trouve dans la variable sum à la fin de l'exécution de l'algorithme.

```
sum ← 0
for (i ← 0 to n − 1 step 1)
{
    sum ← sum + tab[i]
}
```

On peut également écrire une boucle à l'aide de l'instruction **while** qui répète du code tant que sa condition est vraie. Dans l'exemple suivant, on va diviser un nombre entier positif n par 2, puis par 3, ensuite par 4 ... jusqu'à ce qu'il ne soit plus composé que d'un seul chiffre (c'est-à-dire jusqu'à ce que $n < 10$).

```
d ← 2
while (n ≥ 10)
{
    n ← n/d
    d ← d + 1
}
```

Souvent les algorithmes seront dans un cadre et précédés d'explications. Après **Input**, on définit chacun des arguments (variables) donnés en entrée à l'algorithme. Après **Output**, on définit l'état de certaines variables à la fin de l'exécution de l'algorithme et éventuellement la valeur retournée. Une valeur peut être retournée avec l'instruction **return**. Lorsque cette instruction est exécutée, l'algorithme s'arrête et la valeur donnée est retournée.

Voici un exemple en reprenant le calcul de la somme des éléments d'un tableau.

```
Input  : tab[ ], un tableau de n nombres.
         n, le nombre d'éléments du tableau.
Output : sum, la somme de tous les nombres contenus dans le tableau.

sum ← 0
for (i ← 0 to n − 1 step 1)
{
    sum ← sum + tab[i]
}
return sum
```

Remarque: dans ce dernier exemple, la variable i est seulement utilisée comme compteur pour la boucle **for**. Il n'y a donc aucune explication à son sujet, ni dans **Input** ni dans **Output**, et sa valeur n'est pas retournée.