OI 2010 **Finale** 12 Mai 2010	**Remplissez ce cadre en MAJUSCULES et LISIBLEMENT, svp** PRÉNOM : ..... NOM : ..... ÉCOLE :	**Réservé**

Olympiades belges d'Informatique (durée : 1h15 maximum)

Ce document est le questionnaire de **la partie papier** de la finale des Olympiades belges d'Informatique pour **la catégorie supérieur**. Il comporte huit questions qui doivent être résolues en **1h15 au maximum**. Chaque question est accompagnée d'un temps de résolution, donné à titre purement indicatif.

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

1. N'indiquez votre nom, prénom et école **que sur la première page**. Sur toutes les autres pages, vous ne pouvez écrire que dans les **cadres prévus** pour votre réponse.

2. Vous ne pouvez avoir que de quoi écrire avec vous ; les calculatrices, GSM, ... sont **interdits**.

3. Vos réponses doivent être écrites **au stylo ou au bic** bleu ou noir. Pas de réponses laissées au crayon. Si vous désirez des feuilles de brouillon, demandez-en auprès d'un surveillant.

4. Pour les questions de type QCM, vous ne devez choisir qu'**une seule réponse**. Cochez la case de votre choix. Si vous vous êtes trompé, noircissez la case erronée pour annuler votre choix. Une réponse correcte rapporte 1 point, une abstention vaut 0 point et une mauvaise réponse est sanctionnée par $-0{,}5$ point.

5. Vous **devez** répondre aux questions ouvertes en **pseudo-code**. Les erreurs de syntaxe ne sont pas prises en compte pour l'évaluation. Sauf mention contraire, vous ne pouvez utiliser aucune fonction prédéfinie à l'exception de `max (a, b)`, `min (a, b)` et `pow (a, b)` qui calcule a^b.

6. Les tableaux sont indicés de 0 à $n-1$, où n correspond à leurs tailles. La notation **`for`** $(i \leftarrow a$ **`to`** b **`step`** $k)$ indique une boucle qui se répète tant que $i \leq b$ pour i initialisé à a et incrémenté de k à la fin de chaque itération.

7. 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.

8. Il est strictement **interdit de manger ou boire** durant l'épreuve. 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 une heure et quart** pour répondre à toutes les questions.

Bonne chance !

Questionnaire finale papier supérieur

Question 0 – Amuse-bouche (10 min)

1. Soit la fonction `incorrect (n, s)` qui renvoie **`true`** si l'étudiant `s` s'est trompé à la question `n` et **`false`** sinon. Laquelle des expressions suivantes teste-t-elle qu'il n'a aucune réponse correcte parmi les trois premières questions ?

☐	`not incorrect (1, s) or not incorrect (2, s) or not incorrect (3, s)`
☐	`incorrect (1, s) and incorrect (2, s) and incorrect (3, s)`
☐	`incorrect (1, s) or incorrect (2, s) or incorrect (3, s)`
☐	`not (incorrect (1, s) or incorrect (2, s) or incorrect (3, s))`

2. Laquelle des expressions suivantes est-elle équivalente à : `max (`a`, 10) =` b **`and not`** $(a > 2c)$?

☐	**not** `(max (`a`,10) =` b **and** $2c \geq a$`)`
☐	**not** `(max (`a`,10)` $\neq b$ **or** $a > 2c$`)`
☐	**not** `(max (`a`,10)` $\neq b$ **or** $2c \geq a$`)`
☐	**not** `(max (`a`,10)` $\neq b$ **and** $a > 2c$`)`

3. Quelle est la valeur de n après exécution de cet algorithme ?

```
n ← 0
a ← 5
while (a ≤ 2)
{
    n ← a
    a ← a − 1
}
```

☐	0
☐	1
☐	2
☐	5

Question 1 – Et si on comptait en binaire ? (5 min)

La représentation binaire d'un nombre entier positif correspond à une suite de 0 et de 1. Soit une telle suite de la forme $b_{n-1}b_{n-2}\cdots b_2b_1b_0$, avec $b_i \in \{0,1\}$ pour tous les i tels que $0 \leq i < n$. Une telle suite représente le nombre entier suivant :

$$n = \sum_{i=0}^{n-1} b_i \cdot 2^i \quad = \quad b_0 \cdot 2^0 + b_1 \cdot 2^1 + b_2 \cdot 2^2 + \cdots + b_{n-1} \cdot 2^{n-1}$$

L'algorithme suivant permet de calculer la représentation binaire d'un entier positif n donné. La fonction `concat (A, B)` permet de concaténer les deux chaines de caractères `A` et `B` et produit donc une chaine de caractères comme résultat.

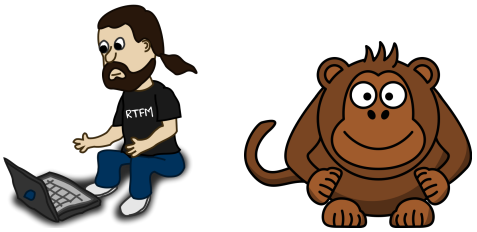
```
Input  : n, entier positif
Output : chaine de caractères représentant n en binaire

convert ← ''                % convert est initialisé avec une chaine de caractères vide

while (n > 0)
{
    if ([...])
    {
        convert ← concat ('0', convert)
    }
    else
    {
        convert ← concat ('1', convert)
    }
    n ← n div 2                                    % div représente la division entière
}

return convert
```

Quelle condition faut-il pour l'instruction **`if`** afin que l'algorithme produise le résultat attendu ? **Attention**, vous ne pouvez utiliser que les opérateurs suivants : $+$, $-$, $*$ et div (division entière).

Q1 **(une condition)**

. . . . . .

Question 2 – Humains et singes sont-ils frères ? (10 min)



Une analyse qui peut être faite sur les séquences d'ADN consiste à comparer deux séquences et calculer la longueur de la plus longue sous-séquence commune. Soient deux séquences $x = x_1, x_2, \cdots, x_m$ et $y = y_1, y_2, \cdots, y_n$. La plus longue sous-séquence commune est une suite d'éléments $z_1, z_2, \cdots, z_k$ apparaissant dans cet ordre dans les deux séquences x et y, sans nécessairement être consécutifs. Prenons par exemple les deux séquences d'ADN suivantes : `x = GATACA` et `y = CGTA`. La plus longue sous-séquence commune est `GTA` et sa longueur est 3.

L'algorithme suivant permet de calculer la longueur de la plus longue sous-séquence commune entre deux chaines de caractères x et y, de longueurs respectives m et n. L'algorithme calcule une matrice c de $m+1$ lignes et $n+1$ colonnes, telle que l'entrée $c[i][j]$ contient la longueur de la plus longue sous-séquence commune entre les chaines de caractères $x[1..i]$ et $y[1..j]$. La notation $x[1..i]$ représente la chaine de caractères x dont on ne garde que les i premiers caractères. Dans notre exemple, $x[1..4]$ représente la chaine de caractères `GATA` et $x[1..0]$ représente la chaine vide.

```
Input  : x, y, deux tableaux de caractères, de longueurs respectives m et n
Output : Une matrice result de m+1 lignes et n+1 colonnes, telle que result[i][j]
         contient la longueur de la plus longue sous-séquence commune entre
         x[1..i] et y[1..j]

result ← matrice d'entiers de m+1 lignes et n+1 colonnes, initialisée avec des 0
                dont les lignes sont indicées de 0 à m et les colonnes de 0 à n

for (i ← 1 to m step +1)
{
    for (j ← 1 to n step +1)
    {
        if (x[i-1] = y[j-1])
        {
            [...]                          % (a)
        }
        else
        {
            c[i][j] = max ([...]);         % (b)
        }
    }
}
```

Quelles instructions manque-t-il à cet algorithme afin qu'il produise le résultat attendu ?

Q2a **(une instruction)**

. . . . . .

Q2b **(deux expressions)**

. . . . . .

Question 3 – Combien de pièces dois-je rendre ? (15 min)

Pour ces vacances, vous avez trouvé un job en tant que caissier dans un supermarché. Un problème auquel sont souvent confrontés les caissiers est de pouvoir rendre la monnaie en pièces. Afin de pouvoir tenir toute la journée, il faut absolument rendre un minimum de pièces à chaque fois. Étant donnés une certaine somme à rendre M et un ensemble de valeurs de pièces $Coins$, l'algorithme suivant va vous permettre de calculer combien de pièces, au minimum, il vous faudra rendre.

```
Input  : M, entier positif, somme d'argent à rendre
         Coins, ensemble d'entiers strictement positifs, valeurs des pièces disponibles
Output : nombre minimal de pièces parmi celles de Coins à rendre pour faire M

tab ← tableau d'entiers de taille M+1, indicé de 0 à M, initialisé avec des 0
tab[0] ← 0

for (m ← 1 to M step +1)
{
    tab[m] ← +∞
    foreach (c ∈ Coins)        % pour chaque dénomination de pièce dans l'ensemble Coins
    {
        if (m ≥ c)
        {
            if ([...])
            {
                tab[m] ← tab[m − c] + 1
            }
        }
    }
}

return tab[M]
```

Quelle condition faut-il pour l'instruction **`if`** afin que l'algorithme calcule le résultat attendu ?

☐	$tab[m]-1 \geq tab[m-1]$
☐	$tab[m-1]+c < tab[m]$
☐	$tab[m] > tab[m-1]$
☐	$tab[m-c]+1 < tab[m]$

Question 4 – Écrire un nombre à l'envers (10 min)

1271384
4831721

Il est possible de partir d'un nombre et d'obtenir un second nombre qui correspond à la lecture à l'envers du premier, rien qu'en effectuant des opérations arithmétiques élémentaires. L'algorithme suivant permet de faire ce calcul et fonctionne pour autant que le nombre original ne se termine pas par un zéro.

```
Input  : n, entier strictement positif, non divisible par 10
Output : nombre qui, si lu à l'envers, est égal à n

result ← 0
while (n ≠ 0)
{
    [...]
    n ← n div 10                                  % div représente la division entière
}

return result
```

Quelle instruction faut-il ajouter afin que l'algorithme calcule le résultat attendu ?

☐	$result \leftarrow 10 \cdot result + n \mod 10$
☐	$result \leftarrow (result + n \mod 10) \cdot 10$
☐	$result \leftarrow n \text{ div } 10 + 10 \cdot result$
☐	$result \leftarrow result \text{ div } 10 + n \mod 10$

(L'opérateur $\mod$ calcule le reste de la division entière.)

Question 5 – À la russe (10 min)

Votre ancien voisin russe vous a laissé, avant de retourner dans son pays natal, un algorithme qu'il vous a garanti pouvoir vous être un jour utile, celui-ci permettant en effet de calculer facilement une opération mathématique fort utile. Il ne vous a malheureusement pas dit de quelle fonction il s'agissait. Voici l'algorithme en question :

```
Input  : a et b, deux entiers strictement positifs
Output : ?

r ← b
while (a ≠ 1)
{
    a ← a div 2
    b ← 2 · b

    if (a mod 2 ≠ 0)
    {
        r ← r + b
    }
}

return r
```

Quelle est la fonction mathématique calculée par cet algorithme ?

Q5 **(une expression mathématique)**

. . . . . .

Question 6 – Trions ! (15 min)

L'algorithme suivant permet de trier les éléments d'un tableau d'entiers de manière croissante. L'algorithme se base sur un tableau temporaire contenant une certaine information sur le tableau à trier. Le remplissage de ce tableau temporaire est manquant et vous devez le compléter.

```
Input  : tab, tableau d'entiers de taille n, indicé de 0 à n − 1
         min, le plus petit élément de tab
         max, le plus grand élément de tab
Output : les éléments du tableau tab sont en ordre croissant

temp ← tableau d'entiers de taille max − min + 1, indicé de 0 à max − min,
       initialisé avec des 0
for (i ← 0 to n − 1 step +1)
{
    [...]
}

j ← 0
for (k ← 0 to max − min step +1)
{
    while (temp[k] ≠ 0)
    {
        data[j] ← min + k
        j ← j + 1
        temp[k] ← temp[k] −1
    }
}
```

Quelle est l'instruction manquante qui permettra à l'algorithme de calculer le résultat attendu ?

Q6 **(une instruction)**

. . . . . .

Question 7 – Sous-tableau de somme maximale (20 min)

Soit un algorithme qui prend en entrée un tableau non-vide d'entiers tab et qui calcule la somme maximale qu'il est possible d'obtenir lorsque l'on prend un sous-tableau non-vide du tableau tab et qu'on fait la somme de ses éléments. Prenons par exemple le tableau $[1, -2, 4]$. Six sous-tableaux sont possibles : $[1]$, $[-2]$, $[4]$, $[1, -2]$, $[-2, 4]$ et $[1, -2, 4]$. Celui dont la somme des éléments est maximale est le troisième ($[4]$), la somme de ses éléments vaut 4.

```
Input  : tab, tableau d'entiers de taille n, indicé de 0 à n − 1, avec n > 0
Output : somme des éléments du sous-tableau non-vide de somme maximale

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

Cet algorithme n'est pas efficace. En effet, pour un tableau de taille n, le temps d'exécution est proportionnel à n^2. Le tableau tab est parcouru beaucoup trop de fois. Il est possible d'écrire un algorithme plus efficace tel que le temps d'exécution soit proportionnel à n plutôt qu'à n^2. On vous demande de compléter l'algorithme suivant dans lequel le tableau n'est parcouru qu'une seule fois.

```
i ← 1
s ← tab[0]
max ← tab[0]
while (i ≠ n)
{
    [...]
}
return max
```

Q7 **(trois instructions)**

. . . . . .

. . . . . .

. . . . . .