

**be-OI 2019**

**Finale - CADETS**  
samedi 30 mars 2019

Remplissez ce cadre en MAJUSCULES et LISIBLEMENT, svp

PRÉNOM : .....  
NOM : .....  
ÉCOLE : .....

O

Réservé

## Olympiade belge d'Informatique (durée : 2h maximum)

Ce document est le questionnaire de la finale de l'Olympiade belge d'Informatique 2019. Il comporte 5 questions qui doivent être résolues en **2h au maximum**.

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

- 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 **cadets**.
  - Pour les élèves en troisième ou quatrième année du secondaire: catégorie **junior**.
  - Pour les élèves de cinquième année du secondaire et plus: catégorie **senior**.
- N'indiquez votre nom, prénom et école **que sur la première page**.
- Indiquez **vos réponses** sur les pages prévues à cet effet, **à la fin du formulaire**.
- Si, suite à une rature, vous êtes amené à écrire hors d'un cadre, répondez obligatoirement **sur la même feuille** (au verso si nécessaire).
- Écrivez de façon **bien lisible** à l'aide d'un **stylo ou stylo à bille** bleu ou noir.
- Vous ne pouvez avoir que de quoi écrire avec vous; les calculatrices, GSM, ... sont **interdits**.
- Vous pouvez toujours demander des feuilles de brouillon supplémentaires à un surveillant.
- Quand vous avez terminé, **remettez la première page (avec votre nom) et les pages avec les réponses**. Vous pouvez conserver les autres.
- 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.
- 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 sponsors et de nos membres:



©2019 Olympiade Belge d'Informatique (beOI) ASBL

Cette oeuvre est mise à disposition selon les termes de la Licence Creative Commons Attribution 2.0 Belgique.

## 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  $\leftarrow$  2002  
age  $\leftarrow$  2019 - 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*. Notez les commentaires dans le code.

```
if (age  $\geq$  18)  
{  
    prix  $\leftarrow$  8 // Ceci est un commentaire.  
}  
else  
{  
    prix  $\leftarrow$  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  $\geq$  18)  
{  
    prix  $\leftarrow$  8 // Prix pour une personne majeure.  
}  
else if (age  $\geq$  6)  
{  
    prix  $\leftarrow$  6 // Prix pour un enfant de 6 ans ou plus.  
}  
else  
{  
    prix  $\leftarrow$  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.

## Question 1 – Pixels-Introduction

Un écran est constitué de pixels. Par exemple un écran Full HD est divisé en 1080 lignes de 1920 pixels chacune. Certains pixels peuvent être défectueux. S'il y en a trop ou s'ils sont mal placés, l'écran doit être rejeté.

Un appareil testeur d'écran retourne un tableau  $pix[ ][ ]$  de  $r \times c$  nombres entiers positifs ou nuls où  $r$  est le nombre de lignes et  $c$  le nombre de colonnes de l'écran testé. Les lignes sont numérotées de haut en bas de 0 à  $r - 1$  et les colonnes sont numérotées de gauche à droite de 0 à  $c - 1$ .

La valeur de  $pix[i][j]$  représente l'état du pixel à l'intersection de la ligne  $i$  et de la colonne  $j$ . Plus cette valeur est haute, mieux le pixel fonctionne. Pour déterminer si un pixel est défectueux, on compare  $pix[i][j]$  à une valeur  $Q$  qui dépend du type d'écran et des exigences de qualité. On considère que le pixel fonctionne correctement si  $pix[i][j] \geq Q$  (autrement dit, le pixel est déclaré défectueux si  $pix[i][j] < Q$ ).

Voici un (petit) exemple présentant les résultats du test d'un écran de 6 lignes et 12 colonnes.

	0	1	2	3	4	5	6	7	8	9	10	11
0	141	101	40	207	129	157	247	244	235	229	108	159
1	175	223	222	254	196	162	152	127	235	243	161	183
2	232	123	226	72	245	109	99	87	254	115	142	0
3	172	123	130	178	155	100	198	226	157	110	202	183
4	112	198	212	241	233	191	218	68	76	72	248	162
5	127	137	110	219	196	194	111	213	141	111	249	147

$pix[ ][ ]$

Si la valeur de qualité  $Q$  est fixée à 100, les 8 pixels en gris seront déclarés défectueux.

Le pixel sur la ligne 0 et la colonne 2 est défectueux car  $pix[0][2] = 40$  est inférieur à  $Q = 100$ .

Avec cette valeur de  $Q$ , 3 lignes et 7 colonnes contiennent des pixels défectueux, il y a 3 pixels défectueux consécutifs dans la ligne 4, mais la ligne avec le plus de pixels défectueux est la ligne 2 qui en contient 4.

Complétez les algorithmes suivants en remplaçant les . . . pour calculer les résultats demandés. Dans chaque algorithme vous pouvez utiliser le tableau  $pix[ ][ ]$  qui contient les résultats donnés par le testeur d'écran, les variables  $r$  et  $c$  qui contiennent le nombre de lignes et de colonnes de l'écran et la variable  $Q$  qui contient la valeur seuil pour la qualité.

**Algorithme Pixels** : Rejette un écran si le nombre de pixels défectueux dépasse une limite donnée.

```

Input  :  $pix[][]$ ,  $r$ ,  $c$  et  $Q$ .
            $dmax$ , le nombre maximal de pixels défectueux que l'on peut accepter.
Output : "XX" si le nombre de pixels défectueux est supérieur à  $dmax$ ,
           "OK" si le nombre de pixels défectueux est inférieur ou égal à  $dmax$ .

 $d \leftarrow 0$ 
for ( $i \leftarrow 0$  to  $r-1$  step 1)
{
  for ( $j \leftarrow 0$  to  $c-1$  step 1)
  {
    if ( ... ) // (a)
    {
       $d \leftarrow \dots$  // (b)
    }
  }
}

if ( ... ) // (c)
{
  return "XX"
}
else
{
  return "OK"
}

```

<b>Q1(a)</b> [2 pts]	<b>Quelle est l'expression (a) dans l'algorithme Pixels ?</b>
----------------------	---

Solution:  $pix[i][j] < Q$

<b>Q1(b)</b> [1 pt]	<b>Quelle est l'expression (b) dans l'algorithme Pixels ?</b>
---------------------	---

Solution:  $d + 1$

<b>Q1(c)</b> [2 pts]	<b>Quelle est l'expression (c) dans l'algorithme Pixels ?</b>
----------------------	---

Solution:  $d > dmax$  (autre solution:  $dmax < d$ )

**Algorithme Lignes** : Compte le nombre de lignes contenant au moins 1 pixel défectueux.

**Input** :  $pix[][]$ ,  $r$ ,  $c$  et  $Q$ .

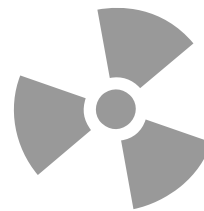
**Output** :  $dr$ , le nombre de lignes contenant au moins un pixel défectueux.

```

dr ← 0
for (i ← 0 to r-1 step 1)
{
  j ← 0
  while (j < c) and (...) // (d)
  {
    j ← ... // (e)
  }

  if ( ... ) // (f)
  {
    ... // (g)
  }
}
return dr

```



Remarque à propos de la manière dont l'ordinateur évalue une expression du type (condition 1) **and** (condition 2) : si (condition 1) est **false**, la (condition 2) n'est pas évaluée. C'est en effet inutile car on sait déjà que l'expression est **false**.

Q1(d) [2 pts]	Quelle est l'expression (d) dans l'algorithme Lignes ?
Solution: $pix[i][j] \geq Q$	
Q1(e) [2 pts]	Quelle est l'expression (e) dans l'algorithme Lignes ?
Solution: $j + 1$	
Q1(f) [2 pts]	Quelle est l'expression (f) dans l'algorithme Lignes ?
Solution: $j < c$ (autre réponse: $j \neq c$ )	
Q1(g) [2 pts]	Quelle est l'instruction (g) dans l'algorithme Lignes ?
Solution: $dr \leftarrow dr + 1$ (autre réponse acceptée: $dr++$ )	

Il faut compter les pixels défectueux dans une zone rectangulaire dont on donne les positions du pixel supérieur gauche (ligne  $r1$ , colonne  $c1$ ) et du pixel inférieur droit (ligne  $r2$ , colonne  $c2$ ).

L'algorithme ci-dessous se contente de parcourir entièrement le rectangle à analyser.

**Algorithme Rectangle** : Compte le nombre de pixels défectueux dans un rectangle.

```

Input  :  $pix[][]$ ,  $r$ ,  $c$  et  $Q$ .
            $r1$  et  $c1$  les numéros de ligne et de colonne du pixel supérieur gauche.
            $r2$  et  $c2$  les numéros de ligne et de colonne du pixel inférieur droit.
            $0 \leq r1 \leq r2 < r$  et  $0 \leq c1 \leq c2 < c$ 
Output :  $d$ , le nombre de pixels défectueux dans le rectangle donné.
 $d \leftarrow 0$ 
for ( $i \leftarrow r1$  to ... step 1)           //(h)
{
    for ( $j \leftarrow \dots$  to ... step 1)     //(i) et (j)
    {
        if ( ... )                             //(k)
        {
            ...                                 //(l)
        }
    }
}
return  $d$ 

```

<b>Q1(h)</b> [1 pt]	<b>Quelle est l'expression (h) dans l'algorithme Rectangle ?</b>
---------------------	--

Solution:  $r2$

<b>Q1(i)</b> [1 pt]	<b>Quelle est l'expression (i) dans l'algorithme Rectangle ?</b>
---------------------	--

Solution:  $c1$

<b>Q1(j)</b> [1 pt]	<b>Quelle est l'expression (j) dans l'algorithme Rectangle ?</b>
---------------------	--

Solution:  $c2$

<b>Q1(k)</b> [1 pt]	<b>Quelle est l'expression (k) dans l'algorithme Rectangle ?</b>
---------------------	--

Solution:  $pix[i][j] < Q$

<b>Q1(l)</b> [1 pt]	<b>Quelle est l'instruction (l) dans l'algorithme Rectangle ?</b>
---------------------	---

Solution:  $d \leftarrow d + 1$

<b>Q1(m)</b> [2 pts]	<b>Combien de fois l'expression (k) sera évaluée si <math>r1 = 500</math>, <math>c1 = 480</math>, <math>r2 = 599</math>, <math>c2 = 511</math> ?</b>
----------------------	--

Solution: 3200 (plus précisément:  $100 \times 32 = 3200$ )

### Question 2 – Pixels-Suite

Cette question utilise les mêmes notions et variables que la question précédente (“Pixel-Introduction”).

Pour comprendre l'énoncé, vous devez d'abord lire celui de "Pixel-Introduction".

Vous pouvez répondre même si vous n'avez pas entièrement résolu la question “Pixel-Introduction”.

Il faut parfois compter successivement les pixels défectueux dans de nombreux rectangles de tailles et de positions différentes sur le même écran.

Dans ce cas, l’**algorithme Rectangle** de la question “Pixel-Introduction” n’est pas efficace; il est possible d’être plus rapide.

Pour cela, il faut utiliser un nouveau tableau  $pixtot[i][j]$  qui permet de donner rapidement (sans aucune boucle **for** ni boucle **while**) le nombre de pixels défectueux dans tout rectangle.

$pixtot[i][j]$  a le même nombre de lignes et de colonnes que  $pix[i][j]$ .

$pixtot[i][j]$  est le nombre de pixels défectueux dans le rectangle allant du pixel supérieur gauche de l'écran (ligne 0, colonne 0) au pixel à l'intersection de la ligne  $i$  et de la colonne  $j$ .

Voici des exemples avec le petit écran présenté dans la question “Pixel-Introduction”.  
(Les couleurs ont été ajoutées pour vous aider à mieux repérer les différentes valeurs dans le tableau.)

	0	1	2	3	4	5	6	7	8	9	10	11
0	141	101	40	207	129	157	247	244	235	229	108	159
1	175	223	222	254	196	162	152	127	235	243	161	183
2	232	123	226	72	245	109	99	87	254	115	142	0
3	172	123	130	178	155	100	198	226	157	110	202	183
4	112	198	212	241	233	191	218	68	76	72	248	162
5	127	137	110	219	196	194	111	213	141	111	249	147

$pix[i][j]$

	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	1	1	1	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	1	1	1	1	1
2	0	0	1	2	2	2	3	4	4	4	4	5
3	0	0	1	2	2	2	3	4	4	4	4	5
4	0	0	1	2	2	2	3	5	6	7	7	8
5	0	0	1	2	2	2	3	5	6	7	7	8

$pixtot[3][6] = 3$

Il y a 3 pixels défectueux dans le rectangle allant du coin supérieur gauche de l'écran jusqu'au pixel en ligne 3 et colonne 6.

	0	1	2	3	4	5	6	7	8	9	10	11
0	141	101	40	207	129	157	247	244	235	229	108	159
1	175	223	222	254	196	162	152	127	235	243	161	183
2	232	123	226	72	245	109	99	87	254	115	142	0
3	172	123	130	178	155	100	198	226	157	110	202	183
4	112	198	212	241	233	191	218	68	76	72	248	162
5	127	137	110	219	196	194	111	213	141	111	249	147

$pix[i][j]$

	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	1	1	1	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	1	1	1	1	1
2	0	0	1	2	2	2	3	4	4	4	4	5
3	0	0	1	2	2	2	3	4	4	4	4	5
4	0	0	1	2	2	2	3	5	6	7	7	8
5	0	0	1	2	2	2	3	5	6	7	7	8

$pixtot[2][7] = 4$

Il y a 4 pixels défectueux dans le rectangle allant du coin supérieur gauche de l'écran jusqu'au pixel en ligne 2 et colonne 7.

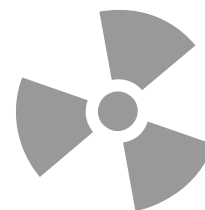


Voici le tableau  $pixtot[ ][ ]$  correspondant à un *autre écran* de 8 lignes et 16 colonnes.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2
2	0	0	0	1	1	1	1	1	1	3	4	5	5	5	5	6
3	0	0	0	1	1	1	1	1	1	3	4	5	5	5	5	6
4	0	0	0	1	1	2	3	3	3	5	6	7	7	7	7	8
5	0	0	0	1	1	2	3	3	3	5	6	7	7	7	7	9
6	0	0	0	1	1	2	3	3	3	6	7	8	8	8	8	10
7	0	0	0	1	1	2	3	3	3	6	7	8	8	8	8	10

$pixtot[ ][ ]$

Utilisez ce dernier tableau pour répondre aux questions suivantes.



<b>Q2(a) [1 pt]</b>	<b>Combien y a-t-il de pixels défectueux sur cet écran ?</b>
Solution: 10	

<b>Q2(b) [1 pt]</b>	<b>Combien y a-t-il de pixels défectueux dans la ligne numéro 2 ?</b>
Solution: 4	

<b>Q2(c) [1 pt]</b>	<b>Quelles lignes n'ont aucun pixel défectueux (donner les numéros séparés par des virgules) ?</b>
Solution: 3, 7	

<b>Q2(d) [1 pt]</b>	<b>Combien de colonnes n'ont aucun pixel défectueux ?</b>
Solution: 9	

Dans les questions qui suivent on utilise les **notations** suivantes :

- “pixel  $(r1, c1)$ ” pour le pixel à l’intersection de la ligne numéro  $r1$  et de la colonne numéro  $c1$ ,
- “rectangle  $(r1, c1) - (r2, c2)$ ” pour le rectangle dont le coin supérieur gauche est le pixel  $(r1, c1)$  et le coin inférieur droit le pixel  $(r2, c2)$ .

<b>Q2(e) [2 pts]</b>	<b>Combien y a-t-il de pixels défectueux dans le rectangle <math>(0, 0) - (6, 6)</math> ?</b>
Solution: 3	

<b>Q2(f) [2 pts]</b>	<b>Combien y a-t-il de pixels défectueux dans le rectangle <math>(3, 3) - (6, 6)</math> ?</b>
Solution: 2	

<b>Q2(g) [2 pts]</b>	<b>Combien y a-t-il de pixels défectueux dans le rectangle <math>(2, 0) - (2, 12)</math> ?</b>
Solution: 3	

<b>Q2(h) [2 pts]</b>	<b>Le pixel (4, 9) est-il défectueux ?</b>
----------------------	--

Solution: <i>NON</i>	
----------------------	--

<b>Q2(i) [3 pts]</b>	<b>Combien y a-t-il de pixels défectueux dans le rectangle (2, 4) – (6, 10) ?</b>
----------------------	---

Solution: 4	
-------------	--

 **Solutions** 

### Question 3 – Arthur, Merlin, le monstre du lac et les autres...

Le Roi Arthur a rapidement besoin de son épée Excalibur pour occire un dragon qui traîne dans le quartier. Merlin le magicien lui explique qu'il y a plusieurs étapes à suivre pour y arriver. Malheureusement, Merlin a la mémoire qui flanche (il a 103 ans !) et n'est plus capable d'expliquer de manière très structurée la marche à suivre. Il se souvient quand même que:

- (a) Les goblins donneront un diamant contre trois pièces d'or.
- (b) La sorcière préparera une potion d'amour contre une pièce de cuivre.
- (c) Le monstre du lac veut une potion d'amour et un bouquet de roses pour séduire une sirène. En échange il donnera Excalibur à Arthur.
- (d) La banque des goblins donnera une pièce d'argent et deux pièces de cuivre contre deux pièces d'or.
- (e) La dame du Lac donnera Excalibur contre un diamant.
- (f) On peut acheter un bouquet de roses pour une pièce d'argent au marché.

Merlin donne **quatre pièces d'or** à Arthur et lui dit qu'il devrait pouvoir obtenir Excalibur à l'aide de ces actions. Il précise qu'Arthur n'est pas obligé de réaliser toutes ces actions, mais que chacune peut être réalisée plusieurs fois si nécessaire.

Arthur commence à réfléchir à la quête qu'il pourrait mener, en commençant avec les quatre pièces d'or que Merlin lui a données. Il envisage plusieurs séquences d'actions. Par exemple: (a), (b), (c), ce qui signifie qu'Arthur commence par échanger trois pièces d'or contre un diamant (il a donc maintenant une pièce d'or et un diamant), puis tente d'acheter une potion d'amour à la sorcière, ce qui est impossible car il n'a pas de pièce de cuivre à ce moment... Arthur ne se décourage pas...

Pour les séquences d'actions suivantes, indiquez en cochant les cases correspondantes sur la feuille de réponses, si elles sont "Impossibles" ou "Faisables". Pour les faisables qui permettent d'obtenir Excalibur, cochez **aussi** la case "Excalibur".

	Impossible	Faisable	Excalibur	Séquences d'actions
<b>Q3(a) [1 pt]</b>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	(d), (f), (b)
<b>Q3(b) [1 pt]</b>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(d), (b), (c), (f)
<b>Q3(c) [1 pt]</b>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(a), (d), (b), (f), (c)
<b>Q3(d) [1 pt]</b>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	(a), (e)
<b>Q3(e) [1 pt]</b>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(d), (a), (e)
<b>Q3(f) [1 pt]</b>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	(d), (d), (b), (f), (b), (b), (f), (c)

<b>Q3(g) [2 pts]</b>	<b>Donnez toutes les séquences d'actions qui permettent d'obtenir Excalibur, en un nombre minimum d'étapes, tout en laissant deux pièces d'or à Arthur.</b>
Solution: (d), (b), (f), (c) et (d), (f), (b), (c)	

Arthur maudit Merlin et sa mémoire défaillante.

Il décide de tracer un dessin dans le sable pour s'aider.

Il distingue les *ressources* (pièces d'or, diamants, potions,...) qu'il représente par des cercles, et les *actions* qu'il représente par des rectangles.

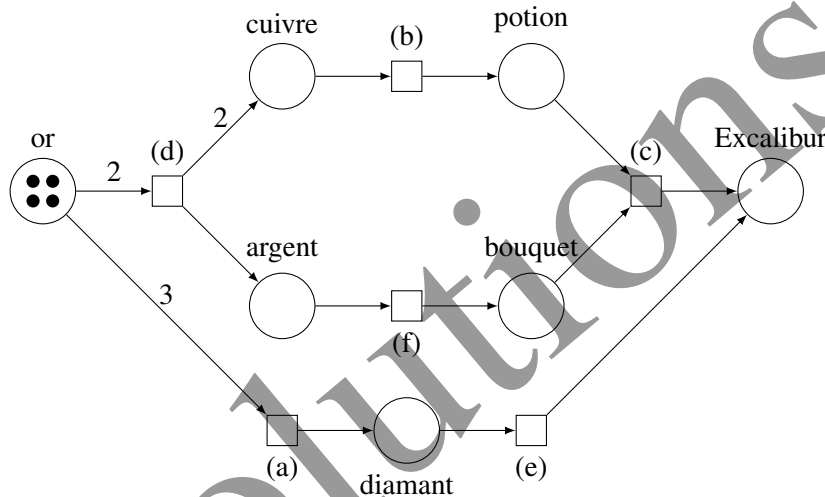
Il pose des cailloux dans les cercles pour compter le nombre de ressources dont il dispose. Il y a par exemple quatre cailloux dans le cercle "pièces d'or" au début de sa quête.

Il trace une flèche d'une ressource *R* vers une action *A* quand *A* a besoin de *R* pour avoir lieu, et indique sur la flèche la quantité de ressources nécessaires. Il y a par exemple une flèche étiquetée 2 allant de "pièces d'or" à (d).

De même, une flèche allant d'une action *A* à une ressource *R* indique que l'action *A* produit cette ressource (et en quelle quantité).

**Pour simplifier, quand les quantités de ressources produites ou consommées sont de 1, Arthur n'indique rien sur la flèche.**

Voici le dessin qu'il obtient:



Par exemple, si Arthur effectue maintenant l'action (d), il ne restera que deux cailloux dans la ressource "or", mais il y en aura deux dans la ressource "cuivre" et un dans la ressource "argent".

De même, pour effectuer l'action (c), il faut *au moins* un caillou dans la ressource "potion" **et** *au moins* 1 caillou dans la ressource "bouquet".

Effectuer cette action (c) a pour effet d'enlever un caillou de ces deux ressources et d'en ajouter un dans "Excalibur", le but de la quête d'Arthur.

Pour effectuer une action, il faut donc que les conditions données par *toutes* les flèches entrantes de l'action soient satisfaites, et cela déclenche les effets donnés par *toutes* les flèches sortant de l'action.

En supposant qu'Arthur commence avec 4 cailloux dans "or" et aucun caillou dans les autres ressources, combien y aura-t-il de cailloux dans chacune des ressources après qu'Arthur aura effectué les séquences d'actions suivantes ?

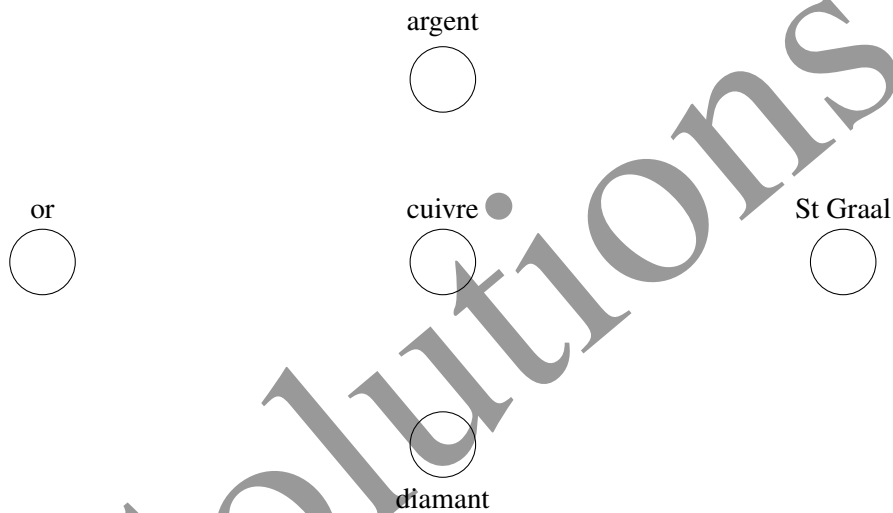
<b>Q3(h) [6 pts]</b>	<b>Après (d), (b), (f) ?</b>
Solution: or: 2 cuivre: 1 argent: 0 potion: 1 bouquet: 1 diamant: 0 Excalibur: 0	

<b>Q3(i) [7 pts]</b>	<b>Après (d), (b), (b), (d), (b), (f) ?</b>
Solution: or: 0 cuivre: 1 argent: 1 potion: 3 bouquet: 1 diamant: 0 Excalibur: 0	

Arthur commence alors une nouvelle quête pour obtenir le Saint-Graal. Voici les nouvelles instructions de Merlin:

- (a) Les gobelins te donneront deux pièces d'argent et une de cuivre contre une pièce d'or.
- (b) Le bijoutier te vendra un diamant au prix de deux pièces d'or.
- (c) Mon vieil ami Gandalf est tellement myope et gâteux qu'il te donnera une pièce d'or pour chaque pièce d'argent que tu lui donneras.
- (d) Les Amazones ont ouvert une nouvelle boutique en ligne où tu pourras acheter le Saint-Graal pour quatre pièces d'argent, une de cuivre et un diamant !

Comme dans la quête précédente, Arthur souhaite avoir un dessin résumant ces possibilités. Il commence par tracer les cercles (ressources) mais a besoin de votre aide pour ajouter les actions et les flèches.



<b>Q3(j) [8 pts]</b>	<b>Ajoutez les actions (a), (b), (c) et (d) et les flèches (avec les quantités) en fonction de la description donnée par Merlin (faites vos essais ci-dessus au brouillon avant de recopier proprement sur la feuille de réponses).</b>
Solution:	

### Question 4 – Nombres au tableau

Alice s’amuse avec des entiers positifs écrits sur un tableau noir. Elle en choisit deux qu’elle efface, puis elle écrit soit leur somme, soit leur différence absolue<sup>1</sup>.

Il y a maintenant un nombre de moins au tableau et Alice recommence ainsi jusqu’à ce qu’il n’en reste qu’un seul.

Alice veut obtenir *la plus petite valeur possible pour le dernier nombre*, elle appelle cette valeur minimum *la réponse*.

Exemple avec 1, 4, 6: Alice pourrait commencer par additionner 1 et 6, ce qui laisse 4 et 7 au tableau qu’elle remplace ensuite par leur différence absolue  $|4 - 7| = 3$ . Mais elle peut faire mieux: commencer par additionner 1 et 4, ce qui laisse 5 et 6 au tableau, puis calculer leur différence absolue  $|5 - 6| = 1$ . Il n’est pas possible de faire mieux, donc *la réponse* que cherche Alice est 1.

<b>Q4(a) [1 pt]</b>	<b>Quelle est la réponse pour la liste 4, 2 ?</b>
Solution: 2	
<b>Q4(b) [1 pt]</b>	<b>Quelle est la réponse pour la liste 2, 6, 3 ?</b>
Solution: 1	
<b>Q4(c) [2 pts]</b>	<b>Quelle est la réponse pour la liste 3, 5, 8, 10 ?</b>
Solution: 0	
<b>Q4(d) [2 pts]</b>	<b>Quelle est la réponse pour la liste 14, 51, 29, 52, 15, 30 ?</b>
Solution: 1	

Alice veut essayer un jeu plus difficile.

Elle se pose des questions comme “Existe-t-il une liste de 4 entiers entre 1 et 10 pour laquelle la réponse est 2 ?”

Dans ce cas la liste 1, 1, 3, 7 fonctionne car  $|7 - (3 + (1 + 1))| = 2$  et qu’il n’est pas possible d’obtenir moins.

Attention, rappelez-vous que *la réponse* est la *plus petite valeur* que l’on peut obtenir à la fin. Par conséquent la liste 1, 2, 3, 4 ne fonctionne pas, car même si  $|(4 + 2) - (1 + 3)| = 2$ , ce n’est pas *la réponse* car il est possible d’obtenir moins en effectuant par exemple  $|(4 + 1) - (3 + 2)| = 0$ .

Dans les questions ci-dessous, on cherche une liste pour laquelle *la réponse* est imposée d’avance. Si vous en trouvez une, écrivez ses éléments séparés par des virgules. Si vous êtes convaincu qu’une telle liste n’existe pas, écrivez “NON”. Une liste peut contenir plusieurs fois le même nombre.

<b>Q4(e) [1 pt]</b>	<b>Existe-t-il une liste de 2 entiers entre 1 et 8 pour laquelle la réponse est 3 ?</b>
Solution: 1,4 ou 2,5 ou 3,6 ou 4,7 ou 5,8	
<b>Q4(f) [2 pts]</b>	<b>Existe-t-il une liste de 3 entiers entre 1 et 6 pour laquelle la réponse est 2 ?</b>
Solution: 1,1,4 ou 1,2,5 ou 1,3,6 ou 2,2,2 ou 2,2,6 ou 2,3,3 ou 2,4,4 ou 2,5,5 ou 2,6,6 ou 3,3,4 ou 3,4,5 ou 3,5,6 ou 4,4,6	
<b>Q4(g) [2 pts]</b>	<b>Existe-t-il une liste de 4 entiers entre 1 et 5 pour laquelle la réponse est 5 ?</b>
Solution: NON	

<sup>1</sup>La différence absolue de deux nombres  $x$  et  $y$  est la valeur absolue de leur différence, notée  $|x - y|$  et obtenue en soustrayant le plus petit nombre du plus grand. Exemples: la différence absolue entre 3 et 5 vaut  $|3 - 5| = |5 - 3| = 2$ , la différence absolue entre 7 et 7 vaut  $|7 - 7| = 0$ .

<b>Q4(h) [2 pts]</b>	<b>Existe-t-il une liste de 5 entiers entre 1 et 3 pour laquelle la réponse est 3 ?</b>
----------------------	---

Solution: 3,3,3,3,3

<b>Q4(i) [2 pts]</b>	<b>Existe-t-il une liste de 6 entiers entre 1 et 2 pour laquelle la réponse est 1 ?</b>
----------------------	---

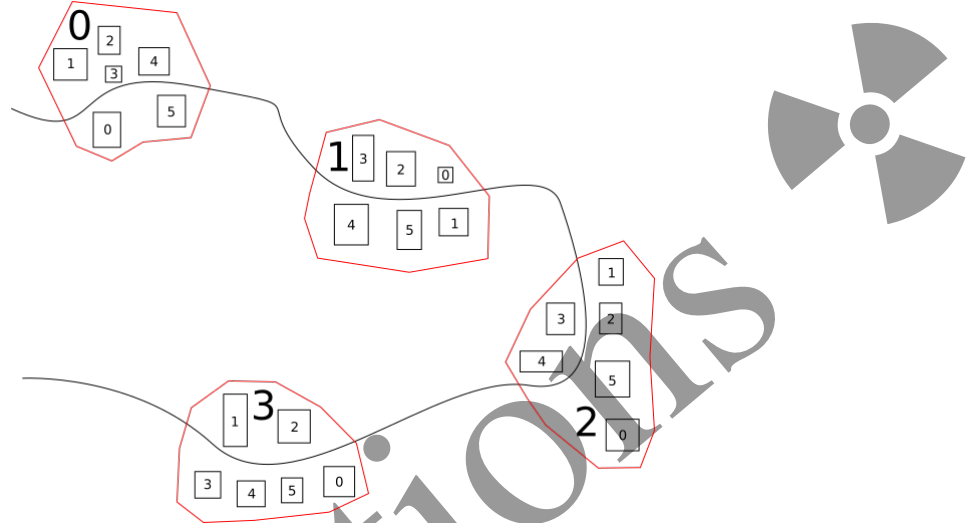
Solution: 1,1,1,1,1,2 ou 1,1,1,2,2,2 ou 1,2,2,2,2,2



### Question 5 – Deux numérotations

Une région montagneuse est constituée de  $n$  villages le long d'une même route. Les villages sont numérotés de 0 à  $n - 1$  et contiennent tous le même nombre  $t$  de maisons. Dans chaque village, on distingue les maisons grâce à un **numéro local** de 0 à  $t - 1$ .

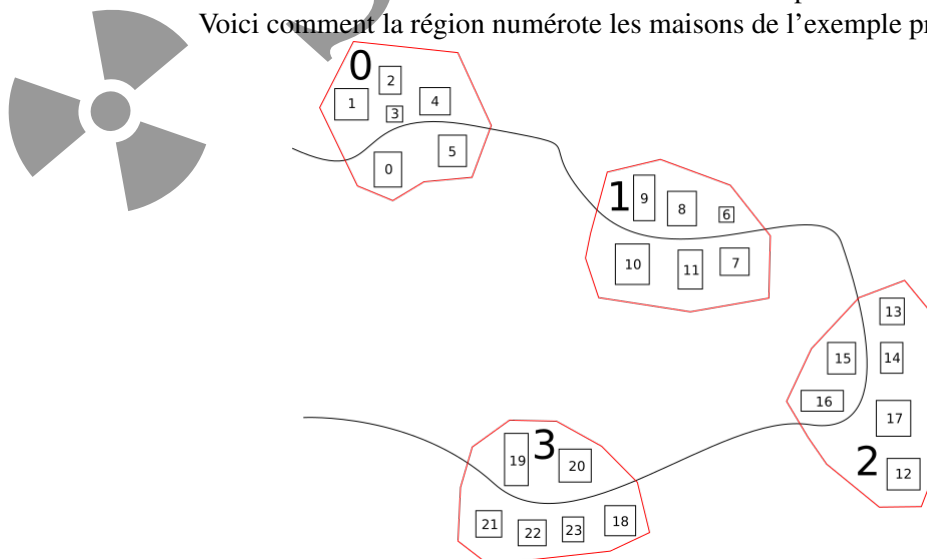
Voici un exemple avec 4 villages et 6 maisons par villages.



Les villages sont en rouge avec leur numéro écrit en grand.  
Chaque maison est représentée par un petit rectangle avec son numéro local.

Chaque maison possède aussi un **numéro régional** donné par la région qui gère l'ensemble des villages. Les numéros régionaux sont attribués en ordre croissant à partir de 0, en parcourant les villages dans l'ordre de leurs numéros et en suivant l'ordre local des maisons à l'intérieur de chaque village. On commence donc par les maisons du village 0, on continue avec celles du village 1, puis celles du village 2,...

Le dessin ci-dessous vous aidera à comprendre.  
Voici comment la région numérote les maisons de l'exemple précédent.



Le **numéro régional** de chaque maison est noté dans le petit rectangle qui la représente.



<b>Q5(a) [3 pts]</b>	<b>S'il y a 7 villages de 10 maisons, quel est le numéro régional de la maison de numéro local 2 dans le village 4 ?</b>
Solution: 42	
<b>Q5(b) [3 pts]</b>	<b>Si les villages ont 13 maisons, quel est le numéro régional de la maison de numéro local 9 dans le village 7 ?</b>
Solution: 100	
<b>Q5(c) [3 pts]</b>	<b>Si les villages ont 25 maisons, quel est le numéro de village et le numéro local de la maison de numéro régional 83 ?</b>
Solution: Village numéro 3 , numéro local 8	
<b>Q5(d) [3 pts]</b>	<b>Si la maison de numéro local 8 dans le village 5 a le numéro régional 63 , quel est le nombre de maisons dans chaque village ?</b>
Solution: 11	
<b>Q5(e) [3 pts]</b>	<b>S'il y a <math>n</math> villages de <math>t</math> maisons, donnez une expression du numéro régional de la maison de numéro local <math>a</math> dans le village <math>b</math>.</b>
Solution: $b \cdot t + a$	

Monsieur Dubois doit visiter toutes les maisons de la région dans un ordre très particulier. Il doit commencer par les maisons dont le numéro local est 0, dans l'ordre des villages, continuer avec celles dont le numéro local est 1, dans l'ordre des villages, et ainsi de suite jusqu'à terminer par la maison de plus grand numéro local dans le dernier village.

Un fonctionnaire de la région doit préparer une liste des numéros régionaux dans l'ordre des visites. Par exemple, avec 4 villages de 6 maisons (comme sur les dessins), la liste est 0, 6, 12, 18, 1, 7, 13, 19, 2, 8, 14, 20, 3, 9, 15, 21, 4, 10, 16, 22, 5, 11, 17, 23.

L'algorithme **Visites** constitue ce genre de liste en tenant compte du nombre de villages et de maisons par village. Il remplit un tableau  $V[]$  (contenant autant d'éléments qu'il y a de maisons dans la région) de telle sorte que pour tout indice  $i$ ,  $V[i]$  est le numéro régional de la maison où Monsieur Dubois doit se rendre lors de la  $i^{\text{e}}$  visite (comme d'habitude, les visites sont numérotées à partir de 0).

#### Algorithme Visites :

```

Input  :  $n$ , le nombre de villages.
            $t$ , le nombre de maisons par village.
            $V[]$ , le tableau initialisé avec des 0.
Output :  $V[]$ , rempli avec les numéros régionaux dans l'ordre des visites.

 $k \leftarrow 0$ 
for ( $i \leftarrow 0$  to  $t-1$  step 1)
{
    for ( $j \leftarrow 0$  to  $n-1$  step 1)
    {
         $V[\dots] \leftarrow \dots$  // ( $f$ ) ( $g$ )
         $k \leftarrow k+1$ 
    }
}

```

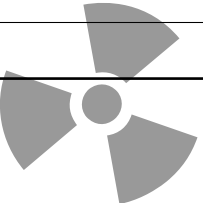
```
}  
return V
```

**Q5(f) [2 pts]** Quelle est l'expression (f) dans l'algorithme Visites ?

Solution:  $k$

**Q5(g) [3 pts]** Quelle est l'expression (g) dans l'algorithme Visites ?

Solution:  $j \times t + i$

  
**Solutions**  
