EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE – LAUSANNE
POLITECNICO FEDERALE – LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY – LAUSANNE

Faculté Informatique et Communications
Cours ICC aux sections MA et PH
Chappelier J.-C.

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

INFORMATIQUE, CALCUL & COMMUNICATIONS

Sections MA & PH

Correction Examen final

20 décembre 2013

SUJET 1

Instructions :

- **VOUS DEVEZ BIEN SÛR FAIRE CES CORRECTIONS DANS LA PLUS STRICTE CONFIDENTIALITÉ.**
  Veuillez en particulier ne *pas* diffuser le corrigé. Merci.
- Veillez à garder l'ordre alphabétique de classement des copies.
- Re-vérifiez bien deux fois vos corrections. Nous faisons tous des erreurs...
- Une fois la correction terminée, remplissez le fichier de notes envoyé par email.
  Mettre explicitement un 0 à tous ceux qui ont rendu une feuille blanche (par opposition à « vide » qui signifiera que vous n'avez pas vu de copie à ce nom).

Exercice 1 – Utilisation d'un ordinateur [3 points]

Question 1.1) Sur un ordinateur avec un processeur à 32 bits, combien de temps faut-il pour lire (en mémoire) un fichier de 32 Mo si le disque transfert 4000 *mots* par ms ?

❑ 0.25 s ❑ 0.25 μs ❑ 16 s ✔ 2 s

Question 1.2) Vous voulez télécharger un fichier de 2 Go depuis un site Web. Votre ordinateur à 4 possibilités d'accéder à Internet : A, B, C et D, de capacité respective 2, 5, 10 et 12 mégabits par seconde. Malheureusement chacune de ces 4 possibilités est servie par un serveur différent ayant en ce moment chacun une capacité respective de 10, 4, 3 et 3 mégabits par seconde.

Quelle connection devez-vous utiliser pour réduire votre temps de téléchargement ?

❑ A ✔ B ❑ C ❑ D

Question 1.3) Un ordinateur avec une mémoire cache de 4 blocs exécute un programme qui utilise 6 blocs de mémoire : A, B, C, D, E et F. Si la mémoire cache a besoin de place, elle renvoie en mémoire le bloc qu'elle n'a pas utilisé depuis le plus longtemps.

Combien de défauts de cache se produisent si le programme accède aux données dans l'ordre :

A, B, A, B, C, D, A, F, A, E, D

✔ 6 ❑ 7 ❑ 8 ❑ 9

Exercice 2 – Multiplication de matrices [3 points]

Question 2.1) On s'intéresse à la multiplication de matrices : $C = A \times B$. Ces matrices sont stockées ligne par ligne en mémoire. Par exemple, si les matrices A et B sont respectivement de taille 5x4 et 4x5, elles sont alors stockées dans la mémoire de la façon suivante :

```
A(1,1) à l'adresse 1048          B(1,1) à l'adresse 4004
A(1,2)             1049          B(1,2)             4005
...                              ...
A(1,4)             1051
A(2,1)             1052
...
A(5,4)             1067          B(4,5)             4023
```

Pour la *lecture* de A, il est préférable d'avoir :

✔ de grands blocs car la localité spatiale des accès est importante.
❑ de grands blocs car la localité temporelle des accès est importante.
❑ de petits blocs car la localité temporelle des accès est importante.
❑ de petits blocs car la localité spatiale des accès est importante.

Question 2.2) Pour effectuer cette multiplication, on utilise l'algorithme naïf suivant :

Multiplication de matrices
entrée : A, B *(nombre de colonnes de A = le nombre de lignes de B)* sortie : $C = A \times B$

Pour i de 1 à **nombre_de_lignes**(A)
  Pour j de 1 à **nombre_de_colonnes**(B)
  $c_{i,j} \longleftarrow 0$
  Pour k de 1 à **nombre_de_colonnes**(A)
  $c_{i,j} \longleftarrow c_{i,j} + a_{i,k} \times b_{k,j}$

Quel est le problème avec cet algorithme s'il est traduit de façon similaire en un programme en assembleur (c'est-à-dire compilé directement) ?

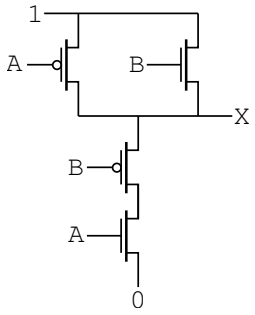
❑ Il n'est pas compatible avec de l'assembleur ; certaines instructions ne peuvent pas être réalisées.
❑ Les deux matrices A et B n'étant pas nécessairement carrées, le résultat n'est pas toujours correct.
✔ L'accès aux éléments de B risque de causer un défaut de cache à chaque itération.
❑ L'accès aux éléments de A se fait dans un ordre qui ne permet pas une bonne localité temporelle.

Question 2.3) Quelle serait une bonne stratégie pour améliorer la traduction de l'algorithme précédent en machine (optimisation du compilateur) ?

❑ Ajouter des lignes ou des colonnes de 0 pour rendre les structures des deux matrices symétriques.
❑ Inverser les boucles i et j.
❑ Représenter les matrices A et B par colonne (au lieu de par ligne).
✔ Représenter la matrice B par colonne, mais laisser la réprésentation de A inchangée.

Exercice 3 – Circuits [2 points]

Question 3.1) Quelle est la table de vérité de la sortie `X` du circuit suivant :



❑

B \ A	0	1
0	0	1
1	1	0

✔

B \ A	0	1
0	1	0
1	1	1

❑

B \ A	0	1
0	1	1
1	0	1

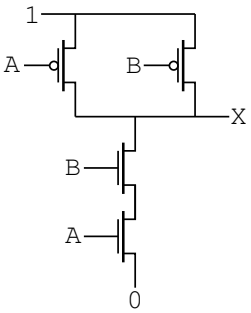
❑

B \ A	0	1
0	1	0
1	0	1

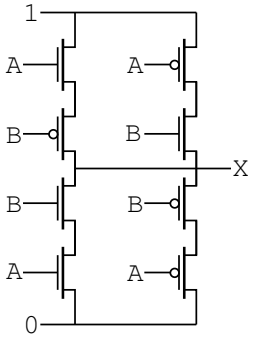
Question 3.2) Quel circuit a pour sortie X suivant la table de vérité suivante :

B \ A	0	1
0	0	1
1	1	0

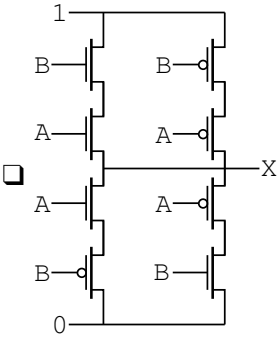
❑



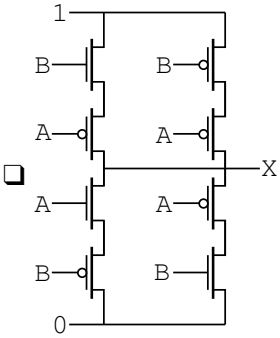
✔



❑



❑



Exercice 4 – Sécurité [2 points]

Question 4.1) Si je veux transmettre vos notes au service académique (SAC) à la fin de l'année, mais souhaite qu'elles restent lisibles (par exemple par vous), alors j'assure :

✔ l'intégrité des notes en calculant leur empreinte cryptographique.
❑ l'authenticité des notes en calculant leur empreinte cryptographique.
❑ l'authenticité des notes en les chiffrant avec la clé publique du SAC.
❑ la confidentialité des notes en les chiffrant avec la clé publique du SAC.

Question 4.2) Dans un système de cryptographie à clé publique,

❑ aucune confidentialité n'est possible puisque la clé est publique.
✔ chaque participant possède deux clés et utilise la clé publique du destinataire pour lui adresser un message confidentiel.
❑ chaque participant possède au moins deux clés et utilise sa propre clé privée pour envoyer des messages confidentiels.
❑ on peut chiffrer les messages mais pas les signer.

Exercice 5 – Code processeur [4 points]

Question 5.1) Lequel de ces constituants *ne* fait *pas* partie de l'architecture de von Neumann :

❑ l'unité arithmétique et logique ❑ la mémoire ❑ l'unité de contrôle ✔ le disque dur

Question 5.2) **[cette question vaut 2 points]**
Voici un code en assembleur, où « `cont_pos x, a` » saute à la ligne `a` si la valeur `x` est *strictement* positive :

```
1: charge   r1, 0
2: charge   r2, 1
3: somme    r3, r1, r2
4: charge   r1, r2
5: charge   r2, r3
6: somme    r0, r0, -1
7: cont_pos r0, 3
```

Quelle est la valeur du registre `r2` lorsque le programme se termine, sachant que le registre `r0` contient au départ la valeur 4 ?

❑ 1 ✔ 5 ❑ 8 ❑ 13

Question 5.3) Lequel de ces algorithmes correspond au code ci-dessus :

❑

algo. A
entrée : n
sortie : m
$s \longleftarrow 0$
$m \longleftarrow 1$
Tant que $n \geq 0$
$m \longleftarrow m + s$
$r \longleftarrow s$
$s \longleftarrow m$
$m \longleftarrow r$
$n \longleftarrow n - 1$

❑

algo. B
entrée : n
sortie : m
$m \longleftarrow 0$
$s \longleftarrow 1$
Tant que $n > 0$
$m \longleftarrow m + s$
$r \longleftarrow s$
$s \longleftarrow m$
$m \longleftarrow r$
$n \longleftarrow n - 1$

✔

algo. C
entrée : n
sortie : m
$s \longleftarrow 0$
$m \longleftarrow 1$
Tant que $n > 0$
$r \longleftarrow m + s$
$s \longleftarrow m$
$m \longleftarrow r$
$n \longleftarrow n - 1$

❑

algo. D
entrée : n
sortie : m
$m \longleftarrow 0$
$s \longleftarrow 1$
Tant que $n \geq 0$
$r \longleftarrow m + s$
$m \longleftarrow r$
$s \longleftarrow m$
$n \longleftarrow n - 1$

Exercice 6 – Limite du code de Shannon–Fano [6 points]

On considère une séquence de lettres composé uniquement de trois lettres dont la distribution de probabilités est la suivante, avec $0 < p < \frac{1}{3}$:

A	B	C
$1-p$	$p/2$	$p/2$

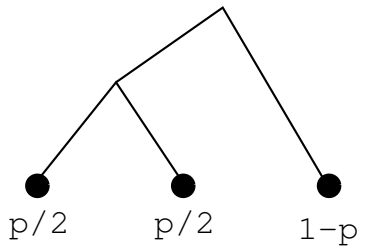
Question 6.1) Quelle est, en fonction de p, l'entropie $H(p)$ de cette séquence de lettres telle que définie en cours ? Mettez la sous la forme $f(p) + p$ (où $f(p)$ est une expression à déterminer).

$$\begin{aligned} H(p) &= (1-p)\log_2 \frac{1}{1-p} + 2 \cdot \frac{p}{2}\log_2 \frac{2}{p} \\ &= (1-p)\log_2 \frac{1}{1-p} + p\left(\log_2 2 + \log_2 \frac{1}{p}\right) \\ &= (1-p)\log_2 \frac{1}{1-p} + p\log_2 \frac{1}{p} + p \end{aligned}$$

BARÈME : 1.5 points pour la formule de l'entropie (dont, si jamais, 1 point pour la définition et 0.5 pour l'appliquer ici). 0.5 point pour la formulation finale qui isole p.

Question 6.2) Quelle est, en fonction de p, la longueur moyenne du code de Shannon-Fano de cette source ? On la notera $L(p)$.

Arbre de code :



$$\begin{aligned} L(p) &= (1-p) \times 1 + \frac{p}{2} \times 2 + \frac{p}{2} \times 2 \\ &= 1 + p \end{aligned}$$

BARÈME : 1.25 point pour l'arbre de code et 0.75 point pour la longueur moyenne.

Question 6.3) On veut montrer que la borne supérieure $H+1$ vue en cours pour la longueur moyenne d'un code de Shannon-Fanon ne peut pas être améliorée.

C'est-à-dire que l'on veut montrer que pour tout $\varepsilon > 0$, il existe toujours une séquence de lettres d'entropie H, dont la longueur moyenne du code de Shannon-Fanon est supérieure à $H + 1 - \varepsilon$.

Considérer pour cela la séquence précédente et calculer la fonction $g(p) = H(p) + 1 - L(p)$.

$$g(p) = (1-p)\log_2 \frac{1}{1-p} + p\log_2 \frac{1}{p} + p + 1 - 1 - p$$

$$= (1-p)\log_2 \frac{1}{1-p} + p\log_2 \frac{1}{p}$$

Quelle est $\lim_{p \longrightarrow 0^+} g(p)$?

$$\lim_{p \longrightarrow 0^+} g(p) = 0$$

Conclure :

Soit $\varepsilon > 0$. De la limite on déduit qu'il existe un p tel que $g(p) < \varepsilon$.

Pour ce p, on peut constuire la source précédente et son code. Ils vérifient alors $H(p) + 1 - L(p) < \varepsilon$, c.-à-d. $H(p) + 1 - \varepsilon < L(p)$.

BARÈME : 0.5 point pour $g(p)$, 1 point pour la limite et 0.5 point pour la conclusion (il suffit de voir qu'il on compris le lien et le problème).

Exercice 7 – Filtres [4 points]

BAREME : 1 point par question.

On considère le signal $X(t)$ suivant :

$$X(t) = 3\,\sin(5\,\pi\,t + \frac{\pi}{3}) + 4\,\sin(4\,\pi\,t + \frac{\pi}{6}) + 5\,\sin(6\,\pi\,t + \frac{\pi}{12})$$

Question 7.1) On applique à ce signal $X(t)$ un filtre passe-bas idéal de fréquence de coupure $f_c = 2.75$ Hz. Quel signal $Y(t)$ obtient-on ?

$$Y(t) = 3\,\sin(5\,\pi\,t + \frac{\pi}{3}) + 4\,\sin(4\,\pi\,t + \frac{\pi}{6})$$

Question 7.2) On s'intéresse au signal $Z(t) = X(t) - Y(t)$. Quelle est sa bande passante ?

3 Hz

Question 7.3) On veut échantillonner $Z(t)$ à une fréquence $f_e = 5$ Hz. Peut-on assurer une reconstruction parfaite ? Si non, que doit on faire ?

Non. Il n'y a pas d'autre solution que d'augmenter la fréquence d'échantillonage au dessus de deux fois la bande passante, soit 6 Hz.

Question 7.4) Finalement, on décide d'échantilloner $Z(t)$ à une fréquence $f_e = 7$ Hz. Quel est le signal $\widehat{Z}(t)$ obtenu après reconstruction ?

$$\widehat{Z}(t) = Z(t)$$

Note : la formule explicite de $Z(t)$ est bien sûr une réponse correcte !

Exercice 8 – Sans répétition [5 points]

On vous demande ici d'écrire un algorithme qui prend en paramètre une liste d'entiers (négatifs, positifs ou nuls) et retourne une autre liste d'entiers ne contenant qu'une et une seule fois chacun des nombres présents dans la liste reçue en entrée.

Par exemple, si la liste reçue en entrée contient les valeurs :

1, 8, 1000, 7, 8, 1, 7, 1000, 1 et 3,

la liste retournée contiendra :

1, 8, 1000, 7 et 3.

Question 8.1) Proposez un algorithme pour la tâche décrite ci-dessus :

Question 8.2) Quelle est la complexité de votre solution ? Justifier votre réponse.

Remarque : ce corrigé est tiré d'un ancien cours de prog. Adaptez le à la donnée...

Je vois au moins cinq idées de solution :

1. ajout au fur et à mesure dans le nouveau tableau que s'il n'y est pas déjà ; i.e. relis à chaque fois depuis le début le nouveau tableau ;
2. recopie tel quel le tableau de départ puis suppression/décalages des doublons ;
3. ajout et parcours d'une structure supplémentaire de même taille que le tableau initial et contenant un booléen qui indique si on garde ou pas la valeur, puis recopie de l'ancien tableau vers le nouveau en fonction de la valeur de ce tableau annexe ;
4. trier la liste dans une liste auxilière, puis la parcourir et ajouter l'élément s'il est différent du dernier élément ajouté. Note : ceci ne préserve par l'ordre, mais ce n'était pas explicitement demandé.
5. utilisation d'une table externe des valeurs indiquant si la valeur est unique ou non.

   Cette solution brutale est coûteuse en mémoire si l'on utilise bêtement un tableau pour les valeurs et peut être améliorée avec des structures de données qu'ils ne devraient pas connaître (e.g. B-trees, tables de hachage). Je ne pense donc pas que cette solution sera proposée.

Note : il n'était **pas** demandé que l'ordre soit conservé. Toute solution correcte mais ne concervant pas l'ordre est donc également juste.

Voici un exemple en C++11 de la première solution :

```
vector<int> detruire_repetitions(vector<int> const& v1)
{
  vector<int> v2;

  for (auto el : v1) {
    // doit-on insérer el dans v2 ?
    bool insere(true);

    // cherche si el est déjà dans v2
    for (size_t j(0); insere and (j < v2.size()); ++j) {
      if (v2[j] == el) { insere = false; }
    }

    // insere el dans v2 si on ne l'a pas trouvé
    if (insere) { v2.push_back(el); }
  }

  return v2;
}
```

Avant tout pour le calcul de la complexité il faut préciser ce qu'est l'entrée et quelle est sa taille, i.e. préciser par rapport à quoi on calcule la complexité. Ici c'est le nombre d'éléments du tableau.

Toutes les solutions sauf les deux dernières sont $\mathcal{O}(n^2)$ car fondamentalement on parcourt, d'une manière ou d'une autre, n fois les i éléments du sous tableau, pour i allant de 1 à n (il faudrait qu'ils expliquent cependant un peu mieux que cela).

L'avant-dernière solution (tri) est en $\mathcal{O}(n \log(n))$.

La dernière solution peut être faite en $\mathcal{O}(n)$ (B-tree ; mais, encore une fois, elle sort du cadre de ce cours).

BARÈME :

① 3 pt pour l'algorithme écrit proprement. Il faut ici noter séparément l'idée de l'algorithme (1 point) et son écriture propre (2 points).

Je vous propose de noter l'idée générale de l'algorithme par tranche de 0.2 points, genre : abordé (0.2), passable (0.4), ok moyen (0.6), ok bien (0.8), très bien (1).

Cette distinction idée/écriture permet de donner des points à ceux qui expliqueraient l'idée sans l'écrire formellement jusqu'au bout. Pour ceux qui ne donnent que l'algorithem proprement écrit, la distinction est bien sur moins claire et la notation doit être plus globale...

② diviser aussi en 5^{e} de points : 0.2 pt pour se poser correctement le problème (par rapport à quoi on calcule, ce qu'est n) et 0.8 pt pour la réponse justifiée (uniquement 0.4 pt sans la justification. En clair qq1 qui dit juste $\mathcal{O}(n^2)$ n'aura que 0.4 pt en tout car il ne positionne pas le problème et ne justifie pas sa réponse).

Exercice 9 – Que fait-il ? [4 points]

Question 9.1) Que fait l'algorithme suivant

devinette
entrée : *Liste* L_1, *Liste* L_2 sortie : *??*

$l_1 \longleftarrow$ **taille**(L_1)
$l_2 \longleftarrow$ **taille**(L_2)
$L_3 \longleftarrow$ liste vide
$i \longleftarrow 1$
Tant que $i \leq l_1$ **ou** $i \leq l_2$
  Si $i \leq l_1$
  Ajouter le i-ième élément de L_1 à L_3
  Si $i \leq l_2$
  Ajouter le i-ième élément de L_2 à L_3
  $i \longleftarrow i + 1$
Sortir : L_3

Donnez une explication claire en français et illustrez par un exemple représentatif. On ne vous demande pas de paraphraser l'algorithme (genre « *On met la taille de* L_1 *dans* l_1, *...* »).

L'algorithme crée une nouvelle liste dont les éléments sont les éléments de L_1 *et* L_2 *jusqu'à épuisement de chacune des listes.*

Par exemple is L_2 *est plus grande que* L_1 *:* $L_{11}, L_{21}, L_{12}, L_{22}, \cdots, L_{1n_1}, L_{2n_1}, L_{2(n_1+1)}, \cdots, L_{2n_2}$.

BARÈME : 2 points en tout : 1 pour voir que l'on parcours les deux listes dans leur entier, et 1 autre pour la construction de L_3.

Question 9.2) On suppose que :
– l'ajout d'un élément à une liste de taille n se fait en $\mathcal{O}(n)$;
– le calcul de la taille d'une liste de taille n se fait en $\mathcal{O}(\log(n))$.
Quelle est la complexité de l'algorithme précédent ? Justifiez votre réponse.

Si n *est la taille de* L_1 *plus celle de* L_2 *(*$n = n_1 + n_2$*), on a* n *parcours de la boucle dans laquelle chaque insertion est* $\mathcal{O}(n_3)$, n_3, *taille de* L_3, *croissant de* 0 *à* $n - 1$.

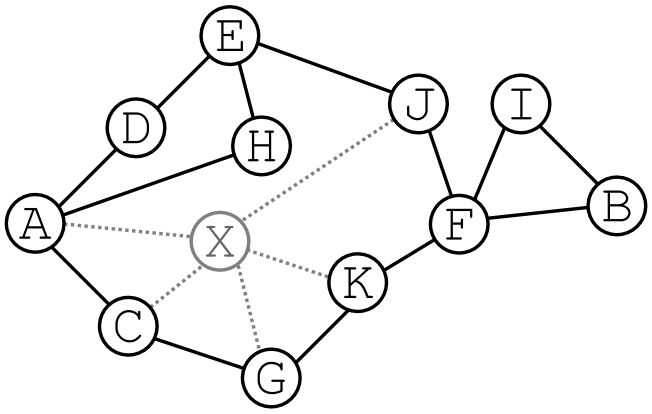
On a donc pour la boucle quelque chose comme : $\sum_{n_3=0}^{n-1} \mathcal{O}(n_3)$, *qui est* $\mathcal{O}(n^2)$.

Par rapport à cela, les appels à ***taille*** *du début sont négligeables (logarithmiques) et donc la compléxité finale est en* $\mathcal{O}(n^2)$.

BARÈME : 2 points en tout : 0.25 pour dire ce qu'est n (somme des 2 tailleS), 1 point pour la réponse et 0.75 pour la justification : 0.5 pour l'explication des boucles et 0.25 pour dire que les log sont négligeables.

Exercice 10 – Routage IP [5 points]

On considère le réseau de routeurs suivant :



Pour simplifier, on se limitera dans tout cet exercice aux tables de routage à distance 2 maximum. Par exemple, le nœud `B` n'indiquera rien sur le nœud `G`.

Question 10.1) Quelle est la table de routage du routeur `K`, *si le nœud* ***`X` n'est pas*** *présent* :

G	1	g
F	1	f
C	2	g
J	2	f
I	2	f
B	2	f

BARÈME : 1.5 point pour la table : 0.25 par ligne, 0.0833 par élément de ligne.

Question 10.2) Quelle est la table de routage du routeur `K`, *si nœud* ***`X` est*** *présent* :

X	1	x
A	2	x
G	1	g
F	1	f
C	2	g
J	2	f
I	2	f
B	2	f

BARÈME : 1.5 point pour la table : 0.1875 par ligne, 0.0625 par élément de ligne.

Question 10.3) Chaque nœud communique à ses voisins les changements de sa table de routage.

Que communique le nœud `K` à ses voisins lorsque le nœud `X` est introduit ?

C'est simplement la différence entre les 2 tables précédentes :

X	1	x
A	2	x

BARÈME : 0.25 points pour la recopie, i.e. comprendre que c'est la différence.

Que communique le nœud `J` à ses voisins lorsque le nœud `X` est introduit ?

X	1	x
A	2	x
C	2	x
G	2	x

BARÈME : 0.75 points (0.1875 par ligne, 0.0625 par élément de ligne)

Question 10.4) A quoi sert d'introduire un tel nœud `X` dans un réseau ?

Cela ajoute de la redondance de route pour rendre le réseau plus résistant aux pannes de nœuds.

Par ailleurs, cela permet de réduire les distances (la latence).

BARÈME : 1 point en tout, 0.5 pour chaque.

Exercice 11 – Comprendre le processeur [4 points]

Question 11.1) Nous avons quatre variables en mémoire aux adresses respectives `@a`, `@b`, `@c` et `@d`.

Que fait le code assembleur suivant, où l'instruction « `décale A, B` » décale la réprésensation mémoire (binaire) de `A` de `B` bits vers la gauche ?

```
 1:  charge     r0, @c
 2:  charge     r1, @a
 3:  multiplie  r2, r1, r0
 4:  décale     r2, 2
 5:  charge     r0, @b
 6:  multiplie  r1, r0, r0
 7:  soustrait  r0, r1, r2
 8:  charge     r1, 1
 9:  cont_equal r0, 0, 13
10:  charge     r1, 0
11:  cont_neg   r0, 13
12:  charge     r1, 2
13:  écrit      r1, @d
```

Calcule (dans `r0`, et en nombre entiers (!)) le(/une valeur entière approchée du) discriminant $\texttt{@b}^2 - 4\texttt{@a}\ \texttt{@c}$*, puis met dans `@d` le nombre de solutions de l'équation du 2nd degré correspondante, i.e. 0 si le discriminant est strictement négatif, 1 s'il est nul et 2 sinon.*

BARÈME : 3 points en tout : 2 pour le calcul de Δ et 1 pour la réponse sur `@d`.

DONNER UN BONUS de 0.5 points s'il y a une remarque sur l'aspect `int` versus `double`.

Question 11.2) A quoi sert l'instruction

```
décale    r2, 2
```

dans le code précédent ?

Expliquez son rôle et illustrez par un exemple.

Elle sert à multiplier par 4 : décaler deux fois à gauche une réprésentation binaire d'un nombre entier le multiplie par 4.

BARÈME : 1 point