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 intermédiaire I

24 octobre 2014

SUJET 1

Instructions :

- Vous disposez d'une heure quinze minutes pour faire cet examen (15h15 - 16h30).
- L'examen est composé de 2 parties : un questionnaire à choix multiples, à 12 points, prévu sur 45 minutes, et une partie à questions ouvertes, à 8 points, prévue sur 30 minutes.
  Mais vous êtes libres de gérer votre temps comme bon vous semble.
- **AUCUN DOCUMENT N'EST AUTORISÉ, NI AUCUN MATÉRIEL ÉLECTRONIQUE**.
- Pour la première partie (questions à choix multiples), chaque question n'a qu'une seule réponse correcte parmi les quatre propositions.
  Indiquez votre réponse en bas de **cette** page en cochant *clairement* une solution parmi les quatre proposées à chaque fois.
  Aucune autre réponse ne sera considérée, et en cas de rature, ou de toute ambiguïté de réponse, nous compterons la réponse comme fausse.
  (Vous êtes autorisés à dégrafer cette page)
- Pour la seconde partie, répondez directement sur la donnée, à la place libre prévue à cet effet.
- Toutes les questions comptent pour la note finale.

Réponses aux quiz :

	A	B	C	D	
question 1 :				✔	1
question 2 :	✔				2
question 3 :			✔		3
question 4 :		✔			4
question 5 :			✔		5
question 6 :				✔	6

	A	B	C	D	
question 7 :			✔		7
question 8 :			✔		8
question 9 :	✔				9
question 10 :				✔	10
question 11 :			✔		11
question 12 :		✔			12

PARTIE QUIZ

1 – FOO, BAR, GO et LENT [2 points]

Question 1) Si le problème « FOO » est dans NP et que l'on sait que « FOO » est au moins aussi difficile que « BAR », quelle affirmation n'est pas correcte ?

A] On peut vérifier une solution à « FOO » en un temps polynomial

B] On peut vérifier une solution à « BAR » en un temps polynomial

C] « BAR » est à coup sûr dans NP

✔**D]** « BAR » ne peut pas être dans P

Question 2) Soit « GO » un problème dans NP et « LENT » un problème dans P prenant le même genre d'entrées (par exemple une liste). On considère le problème « GO_ET_LENT » qui consiste à résoudre « GO » puis « LENT » sur une même entrée. Que peut-on affirmer de « GO_ET_LENT » ?

✔**A]** rien du tout (aucune des trois autres propositions)

B] il est dans NP mais pas dans P

C] il n'est pas dans NP

D] il est dans P

2 – Complexités comparées [2 points]

Question 3) Quelle est la complexité de l'algorithme suivant :

F
entrée : n, *entier naturel* sortie : *une valeur...*

Si $n \leq 2$
  sortir : 1
Sinon
  sortir : $\mathbf{F}(n-1) + \mathbf{F}(n-1)$

A] $\mathcal{O}(n)$ mais pas $\mathcal{O}(1)$

B] $\mathcal{O}(n-1)$ mais pas $\mathcal{O}(n-2)$

✔**C]** $\mathcal{O}(2^n)$ mais pas $\mathcal{O}(n^2)$

D] $\mathcal{O}(n^2)$ mais pas $\mathcal{O}(n)$

Question 4) Est-ce que la réponse change si l'on remplace la dernière expression par « **sortir :** $2\times\mathbf{F}(n-1)$ » ?

A] non cela ne change rien à la complexité

✔**B]** oui cela simplifie grandement la complexité

C] oui cela augmente grandement la complexité

D] oui cela change mais de façon peu significative

suite au dos ☞

3 – Manipulation de nombres entiers [4 points]

On s'intéresse ici à des nombres entiers représentés sur 8 bits.

Question 5) En représentation non signée, à quel nombre entier positif correspond le schéma binaire `11010010` ?

A] 46 **B]** 75 ✔**C]** 210 **D]** 194

Question 6) Toujours pour le même schéma binaire `11010010`, à quel nombre entier signé cela correspondrait-il ?

A] l'opposé de `00101101`, c.-à-d. -210
B] l'opposé de `01010010`, c.-à-d. -82
C] l'opposé de `00101101`, c.-à-d. -45
✔**D]** l'opposé de `00101110`, c.-à-d. -46

Question 7) Si l'on interprète les schémas binaires comme des nombres uniquement *positifs*, quel est le schéma résultant (sur 8 bits) de l'addition de `10001101` et `11001101` ?

A] `01011000` **B]** `101101010` ✔**C]** `01011010` **D]** `11011010`

Question 8) Même question, mais si l'on interprète les schémas binaires comme des nombres *signés* (positifs ou négatifs) :

A] `10100110` **B]** `101101010` ✔**C]** `01011010` **D]** `00100111`

4 – Algorithmes [2 points]

Question 9) Quelle est la complexité de l'algorithme de multiplication vu en série d'exercices et rappelé ici :

multiplication
entrée : a, b *deux entiers naturels non nuls*
sortie : $a \times b$

$x \longleftarrow a$
$y \longleftarrow b$
$z \longleftarrow 0$
Tant que $y \geq 1$
  Si y est pair
    $x \longleftarrow 2\,x$
    $y \longleftarrow y/2$
  Sinon
    $z \longleftarrow z + x$
    $y \longleftarrow y - 1$
sortir : z

✔**A]** $\mathcal{O}(\log(b))$ mais pas $\mathcal{O}(1)$
B] $\mathcal{O}(a\,b)$ mais pas $\mathcal{O}(\log(a\,b))$
C] $\mathcal{O}(b\log(b))$ mais pas $\mathcal{O}(b)$
D] $\mathcal{O}(b)$ mais pas $\mathcal{O}(\log(b))$

Question 10) En supposant que :

— **taille**(L) retourne le nombre d'éléments d'une liste L ;
— **début**(L) retourne le premier élément d'une liste L ;
— **fin**(L) retourne la liste L sans son premier élément ;
— $L[i]$ retourne le i-ième élément d'une liste L ;
— toutes ces opérations ont une complexité temporelle en $\mathcal{O}(1)$ par rapport à la taille de la liste ;

considérez les algorithmes suivants :

algo1

entrée : *liste L non vide*
sortie : *? ?*

```
n ⟵ taille(L)
x ⟵ début(L)
Pour tout e de fin(L)
 Si e < x
  x ⟵ e
sortir : x
```

algo2

entrée : *liste L non vide*
sortie : *? ?*

```
n ⟵ taille(L)
Si n > 1
 Pour i de 1 à n − 1
  Si L[i] < L[i + 1]
   x ⟵ L[i + 1]
   L[i + 1] ⟵ L[i]
   L[i] ⟵ x
sortir : L[n]
```

algo3

entrée : *liste L non vide*
sortie : *? ?*

```
n ⟵ taille(L)
Si n = 1
 sortir : début(L)
Sinon
 x ⟵algo3(fin(L))
 Si début(L)< x
  sortir : début(L)
 Sinon
  sortir : x
```

aide

entrée : *liste L, valeur x*
sortie : *? ?*

```
n ⟵ taille(L)
Si n = 0
 sortir : x
Sinon
 Si x <début(L)
  sortir : aide(fin(L), x)
 Sinon
  sortir : aide(fin(L), début(L))
```

algo4

entrée : *liste L non vide*
sortie : *? ?*

```
sortir : aide(fin(L), début(L))
```

et indiquez laquelle des propositions suivantes est correcte :

A] algo1 a une plus grande complexité que algo3

B] algo2 fait moins de comparaisons que algo1

C] algo2 a une complexité moindre que les trois autres

✔**D]** algo2 et algo4 retournent le même résultat

suite au dos ☞

Question 11) Que valent x, y et z après l'exécution de la portion d'algorithme suivante :

```
x ⟵ 1
y ⟵ 10
z ⟵ 3
Tant que x ≤ 11
 Si x est pair
  x ⟵ x + 3
  y ⟵ 2 y − z
 Sinon
  x ⟵ x + 2
  y ⟵ x − y
  z ⟵ y − 4 + x
```

A] 11, -3 et 4 **B]** 13, -38 et -29 **✔C]** 13, 16 et 25 **D]** 11, -7 et -8

Question 12) Que calcule l'algorithme suivant :

```
devinette
entrée : a et b, entiers naturel
sortie : ? ? ?
  x ⟵ a
  y ⟵ b
  z ⟵ 1
  Tant que y ≥ 1
   Si y est impair
    z ⟵ z × x
   x ⟵ x × x
   y ⟵ y/2   (division entière, quotient)
  sortir : z
```

A] $a \times b$ **✔B]** a^b **C]** $a + b$ **D]** $\log_2(a \times b)$

PARTIE EXERCICES

5 – Ecrire un algorithme et des nombres [4 points]

Question 13) Ecrivez un algorithme qui prend en entrée une liste non vide de mots en majuscules (avec répétitions possibles) et retourne le mot en majuscules le plus fréquent dans cette liste.

Par exemple avec (BOA, ABRI, LOUP, CHAT, CHAT, BOA, CHAT, CHAT, LOUP, CHAT, ABRI) en entrée, l'algorithme retourne « CHAT ».

Il existe plusieurs solutions possibles. En voici quelques unes :

plus fréquent
entrée : L *liste non vide*
sortie : *élément e plus fréquent dans* L

```
trier L
courant ⟵ L[1]
compte ⟵ 1
e ⟵ courant
retenu ⟵ compte
Pour i de 2 à taille(L)
  Si L[i] = courant
    compte ⟵ compte + 1
  Sinon
    Si retenu < compte
      e ⟵ courant
      retenu ⟵ compte
    courant ⟵ L[i]
    compte ⟵ 1
Si retenu < compte
  e ⟵ courant
sortir : e
```

plus fréquent
entrée : L *liste non vide*
sortie : *élément e plus fréquent dans* L

```
t ⟵ taille(L)
retenu ⟵ 0
Tant que L non vide
  compte ⟵ 1
  courant ⟵ L[1]
  retirer L[1] de la liste L
  t ⟵ taille(L)
  Pour j de 1 à t
    Si courant = L[j]
      compte ⟵ compte + 1
      retirer L[j] de la liste L
      t ⟵ t − 1
      j ⟵ j − 1 // pour ne pas augmenter j
  Si retenu < compte
    e ⟵ courant
    retenu ⟵ compte
sortir : e
```

plus fréquent
entrée : L *liste non vide*
sortie : *élément e plus fréquent dans* L

```
t ⟵ taille(L)
retenu ⟵ 0
C ⟵ liste vide
Pour i de 1 à taille(L)
  ajouter 0 à C
Pour i de 1 à taille(L)
  C[i] ⟵ C[i] + 1
  Pour j de i+1 à taille(L) // convention : pas de boucle si i =taille(L)
    Si L[i] = L[j]
      C[i] ⟵ C[i] + 1
      C[j] ⟵ C[j] + 1
  Si retenu < C[i]
    e ⟵ L[i]
    retenu ⟵ C[i]
sortir : e
```

Question 14) De nos jours, comment représente-t-on sur 8 bits la valeur entière -25 dans un ordinateur ?
Justifiez votre réponse.

Réponse : `1110 0111`

Pensez à justifier/expliquer votre réponse, cela vaut (au moins) la moitié des points.

Question 15) Quelle valeur obtient-on si on lui ajoute la valeur binaire `11010101` ?
Exprimez votre réponse en décimal et justifiez la.

Réponse : -68

Pensez à justifier/expliquer votre réponse, cela vaut (au moins) la moitié des points.

suite au dos ☞

6 – Comprendre un algorithme [4 points]

Considérez l'algorithme suivant :

Mystère
entrée : *Une liste L de nombres et un nombre x* sortie : *? ? ?*
$n \longleftarrow$ **taille**(L) $a \longleftarrow 0$ **Pour** i de n à 1 (en décroissant) $\quad a \longleftarrow a \times x + L[i]$ **sortir :** a

Question 16) Que calcule cet algorithme ?

Il **ne** s'agit **pas** ici de paraphraser l'algorithme mais bien de montrer *conceptuellement* ce qui est calculé par cet algorithme. Une formule mathématique explicite/développée peut suffire (mais *une* phrase en français aussi).

Réponse : calcule la valeur du polynôme $\sum_{i=1}^{n} L[i]\, x^{i-1}$

Question 17) En supposant que :
- la complexité de la multiplication de deux nombres de taille s (i.e. écrits sur s bits) est en $\mathcal{O}\left(s \log^2(s)\right)$;
- celle de l'addition $\mathcal{O}(s)$;
- le calcul de la taille d'une liste et l'accès à un de ses éléments ($L[i]$) se font chacun en $\mathcal{O}(1)$;

quelle est la complexité temporelle de cet algorithme ?
Justifiez votre réponse.

Réponse : La réponse est : $\mathcal{O}\left(n\, s \log^2(s)\right)$ où n est la taille de L et s le nombre de bits de x.

(et n'oubliez pas de justifier/expliquer votre réponse !)

Note : il n'y a aucune raison de supposer de lien entre n et s, car même si en effet le plus grand facteur est de l'ordre de x^n, on peut raisonnablement au niveau de cet examen faire l'hypothèse de ne pas le représenter dans toute sa précision, mais uniquement sur s bits (par exemple en virgule flottante) vu que l'on a pas précisé la nature de x (nombre réel ou non).

Si l'on veut aller plus loin [1] et faire l'hypothèse que l'on garde autant de précision (absolue) pour le plus grand facteur, s devient alors la taille utilisée pour représenter x^n, dont on ne sait pas grand chose pour des nombres réels.

Enfin, si l'on fait l'hypothèse que x est un nombre entier et que l'on veut une représentation exacte, alors le « s » utilisé (ce n'est plus le même) devient en $\mathcal{O}(n \log(x))$ et la réponse serait $\mathcal{O}\left(n^2 \log(x) \log^2(n \log(x))\right)$.

1. mais ce n'était pas attendu et n'aurait pas de sens pour des nombres à virgule flottante puisque justement c'est l'erreur *relative* qui est majorée dans de telles représentations !