université PARIS DIDEROT PARIS 7

EXAMEN PF1 12/12/2017 12^{00}–14^{00}

Aucun document ou support autre que le sujet ou les copies d'examen n'est autorisé.
(la copie ou les brouillons du voisin ne sont pas des supports autorisés).
Éteignez impérativement vos mobiles.

Lorsque des calculs sont nécessaires, il est **impératif** de les présenter sur la feuille d'examen. Il est aussi nécessaire de **justifier** ses réponses.

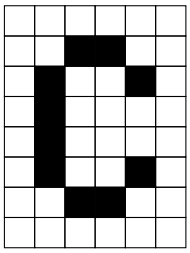
1 Exercice (arithmétique)

Soit les nombres $n_1 = (93)_{10}$ et $n_2 = (-23)_{10}$

1. Donner leurs représentations en complément à deux sur 12 bits ;
2. Donner leur écriture en base 4 et 8.
3. Calculer leur somme en base 2.
4. Calculer le produit de n_2 par 5 en base 2.
5. Donner la valeur du nombre non signé écrit en base 2 : $\overbrace{1\cdots1}^{n}$.
6. Donner la valeur du nombre écrit en complément à deux pour la représentation en taille fixe de longueur n : $\overbrace{1\cdots1}^{n}$.

2 Exercice (codage, image)

Soit la représentation, en image noir et blanc, de la lettre C :



On rappelle que le principe du codage RLE (ici appliqué à des suites de 0 et 1) consiste à coder une suite de symboles identiques par leur longueur, par exemple `0000000111` sera codé par 7, 3. Dans ce codage on considère que l'on commence toujours par des 0, donc le codage du mot `110000` sera codé 0, 2, 4. Dans la suite on supposera que les dimensions de l'image sont connues (ici 6x8).

1. Sans codage combien de bits faut-il pour représenter l'image ? On utilisera 0 pour représenter la couleur blanche et 1 pour la couleur noire.
2. Utiliser le codage RLE pour coder l'image en codant la suite de bits en parcourant l'image en « escargot », en partant du point en haut à gauche, en se déplaçant vers le point à droite, puis en descendant, puis en revenant à gauche, etc. Comme ci-après :

1	2	3
8	9	4
7	6	5

   Fournir la suite des valeurs du codage RLE de l'image de la lettre C en utilisant ce parcours.
3. Si l'on code les entiers de la suite comme des mots binaires de taille fixe, quelle sera la taille du code ? Quelle sera la taille du codage de la suite (c'est-à-dire de l'image) ? Dans notre cas, ce codage permet-il de compresser ?
4. À partir de la suite RLE, donner le tableau associant à chaque valeur y apparaissant, son nombre d'occurrences (c'est-à-dire le nombre de fois que cette valeur est répétée).

5. À partir du tableau construire l'arbre de Huffman correspondant. Prendre soin de préciser les poids de chaque nœud, les valeurs de la suite associées aux feuilles et les étiquettes associées aux arcs.
6. À partir de cet arbre, coder la suite avec le code de Huffman. Quelle est la taille du codage de l'image ? Dans ce cas, le double codage RLE puis Huffman permet-il de compresser ?

3 Exercice (codage, contrôle)

Le code de Luhn est utilisé pour vérifier l'intégrité d'une suite de chiffres ($c_{n-1}...c_0$). Il est utilisé dans de très nombreux cas réels : cartes bancaires, numéro SIRET, etc. Son principe repose sur la construction d'un chiffre de contrôle (C) qui est alors ajouté en fin de la liste d'origine afin de constituer le message final à transmettre ($c_{n-1}...c_0C$).
Le chiffre de contrôle de Luhn est construit de la façon suivante :

— On transforme la suite $c_{n-1}...c_0$ en doublant tous les chiffres de rang pair (c_0, c_2, etc). Lorsque le double d'un chiffre dépasse 10 on fait la somme de ses chiffres (ex. : $7 \rightarrow 14 \rightarrow 5$).

— Ensuite on additionne tous les chiffres obtenus pour obtenir la somme s.

— Le chiffre de contrôle est alors 10-(s/10).

Vérifier l'intégrité consiste à vérifier que le chiffre de contrôle du message correspond bien à la suite qui le précède.
Ex. : Le message 133 est valide mais pas 255.

1. Calculer le code de Luhn pour les suites 123321 et 9876.
2. Vérifier l'intégrité des messages 14373 et 14733.
3. Écrire une fonction Java permettant de calculer le code de Luhn étant donnée un tableau d'entiers : `int luhn(int []c)`. On pourra utiliser une table prédéfinie pour le « doublement » des chiffres... (ainsi que dans la question suivante).
4. Écrire une fonction Java permettant de vérifier qu'un message de Luhn est intègre : `int isLuhn(int []m)`.

4 Exercice (logique, circuit)

Note : dans cet exercice les seules portes autorisées sont ET, OU et NON.
Un industriel souhaite réaliser un nouveau circuit. Ce circuit sera baptisé **low-bit-finder** et il permettra de calculer à partir d'un nombre e représenté en binaire sur 4 bits et sans signe $e_3e_2e_1e_0$:

— une variable booléenne V (Valid) qui vaudra 1 si les entrées ne sont pas toutes nulles et 0 sinon ;

— un nombre binaire de taille fixe n_1n_0 qui représentera l'indice du 1 de plus faible poids dans l'écriture binaire de e (c'est-à-dire l'indice du 1 le plus à droite dans l'écriture binaire de e).

1. Que valent V, n_1, n_0 si l'entrée du circuit code le nombre 12 ?
2. Donner la table de vérité pour les trois fonctions V, n_1 et n_0 ;
3. Donner un circuit, le plus petit possible, qui permet de calculer V (vous êtes autorisés à utiliser des portes avec un nombre d'entrées quelconque) ;
4. Donner un circuit n'utilisant **que des portes à deux entrées** et qui calcule le plus vite possible la fonction V (circuit de plus petite profondeur) ;
5. Donner sous la forme normale disjonctive l'écriture de la formule correspondant aux fonctions n_1 et n_0.
6. À l'aide de tableaux de Karnaugh, simplifier les deux fonctions n_1, n_0.
7. Dessiner le circuit qui calcule n_1 et n_0 à l'aide de portes à deux entrées uniquement.