ICC-T Théorie

Question T1. On considère l'algorithme suivant:

algo_mystère1

entrée : *liste L de n nombres entiers positifs*
sortie : *???*

$m \longleftarrow \lfloor \frac{n}{2} \rfloor$
Pour *i allant de 1 à m*
    Si $L(i) > L(n+1-i)$
        Permuter $L(i)$ *et* $L(n+1-i)$
Sortir : L

A la fin de l'exécution de cet algorithme:

(A) La liste L est triée dans l'ordre croissant.

(B) Les m premiers nombres de la liste L sont tous plus petits ou égaux aux $n-m$ derniers.

(C) La liste L sort inchangée si et seulement si $L(i) = L(n+1-i)$ pour tout $i \in \{1, \ldots, n\}$.

(D) Aucune des affirmations ci-dessus n'est vraie.

Question T2. On considère l'algorithme suivant:

algo_mystère2

entrée : *nombre entier strictement positif n*
sortie : *???*

$s \longleftarrow 0$
Pour *i allant de 1 à n*
    $j \longleftarrow 0$
    $k \longleftarrow 0$
    Tant que $j \leq n$
        $k \longleftarrow k+1$
        Si *k est divisible par i*
            $j \longleftarrow j+1$
    $s \leftarrow s+k$
Sortir : s

Quelle est la complexité temporelle de cet algorithme, dans le pire des cas ?

(A) $\Theta(n)$ (B) $\Theta(n^2)$ (C) $\Theta(n^3)$ (D) plus que polynomiale en n

Question T3. On considère l'algorithme suivant:

algo_mystère3
entrée : *nombre entier strictement positif* n sortie : *???*

Si $n = 1$ *ou* $n = 2$
  Sortir : *1*
$s \longleftarrow 0$
Pour i *allant de 1 à* $n - 2$
  $s \longleftarrow s +$ **algo_mystère3**(i)
Sortir : s

Pour une valeur de $n \geq 3$ donnée, que vaut la sortie de cet algorithme ?

(A) 2^{n-2} (B) 2^{n-3} (C) $F(n-1)$ (D) $F(n-2)$

Note: On rappelle ici que $F(n)$ désigne le n^e nombre de la suite de Fibonacci définie par $F(1) = 1$, $F(2) = 1$, $F(n) = F(n-1) + F(n-2)$ pour $n \geq 3$.

Question T4. Quelle est la complexité temporelle d'**algo_mystère3**(n), dans le pire des cas ?

(A) $\Theta(n)$ (B) $\Theta(n^2)$ (C) $\Theta(n^3)$ (D) plus que polynomiale en n

Question T5. Si $x = 00000110$ dans la représentation binaire des nombres entiers positifs sur 8 bits, alors le calcul de x^3 donne

(A) 01100000 (B) 10110110 (C) 11011000 (D) un overflow

Question T6. Soient f_1, f_2 deux fréquences telles que $f_1 > 2f_2 > 0$. On considère le signal X donné par

$$X(t) = \cos(2\pi(f_1 + f_2)\, t) \cdot \cos(2\pi(f_1 - f_2)\, t), \quad t \in \mathbb{R}$$

Si on échantillonne ce signal à une fréquence f_e, quelle condition *minimale* doit respecter f_e pour que le signal puisse être reconstruit parfaitement à partir de sa version échantillonnée ?

(A) $f_e > 4f_1$ (B) $f_e > 2f_1$ (C) $f_e > 2(f_1 + f_2)$ (D) $f_e > 2f_2$

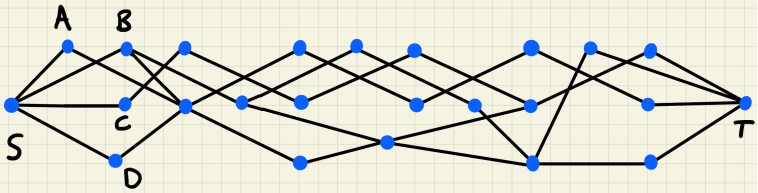
Question T7. Supposons maintenant que le signal X ci-dessus soit filtré par un filtre passe-bas idéal de fréquence de coupure $f_c = f_1$. Quelle sera la sortie $\widehat{X}$ de ce filtre ?

(A) $\widehat{X}(t) = \frac{1}{2} \cdot \cos(4\pi f_1 t)$ (B) $\widehat{X}(t) = \frac{1}{2} \cdot \cos(4\pi f_2 t)$

(C) $\widehat{X}(t) = X(t)$ (D) $\widehat{X}(t) = \cos(2\pi(f_1 - f_2)t)$

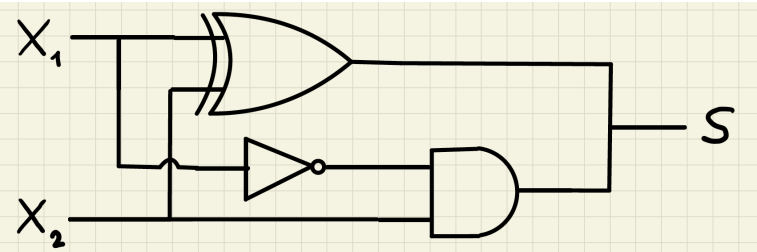
Rappel utile pour les deux questions ci-dessus: Pour tous $a, b \in \mathbb{R}$, on a: $\cos(a+b) = \cos(a) \cdot \cos(b) - \sin(a) \cdot \sin(b)$ et $\cos(a - b) = \cos(a) \cdot \cos(b) + \sin(a) \cdot \sin(b)$.

Question T8. Dans la table de routage du noeud S, quelle est la direction indiquée pour envoyer un paquet à la destination T ? (indiquer simplement la lettre correspondante A, B, C ou D)



Note: On suppose que deux noeuds reliés par une arête sont à distance 1 l'un de l'autre.

Question T9. Le circuit ci-dessous est problématique, car il provoque un court-circuit pour une paire d'entrées (X_1, X_2). Laquelle ?



(A) $X_1 = X_2 = 0$ (B) $X_1 = 1$, $X_2 = 0$ (C) $X_1 = 0$, $X_2 = 1$ (D) $X_1 = X_2 = 1$

Question T10. Supposons qu'Alice et Bob, qui partagent une clé secrète $K = (K_a, K_b)$ d'une longueur de $2n$ bits, utilisent le système DES partiel suivant pour communiquer de manière cryptée:

1. Pour envoyer le message $M = (M_a, M_b)$, Alice effectue successivement les deux opérations suivantes: $C_a = M_a \oplus f(K_a, M_b)$ et $C_b = M_b \oplus f(K_b, C_a)$, où $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ est une fonction non-linéaire donnée (connue de tout le monde). Puis Alice envoie le message chiffré $C = (C_a, C_b)$ à Bob.
2. Pour déchiffrer le message provenant d'Alice, Bob effectue successivement les deux opérations suivantes: $D_b = C_b \oplus f(K_b, C_a)$ et $D_a = C_a \oplus f(K_a, D_b)$,

Supposons maintenant qu'Eve intercepte le message chiffré $C = (C_a, C_b)$, et qu'elle ait également eu accès par un moyen détourné à la première partie de la clé secrète K_a. Laquelle des affirmations suivantes est vraie ?

(A) Eve peut décrypter sans problème le message entier M.

(B) Eve peut décrypter sans problème la première partie M_a du message M, mais pas la seconde partie M_b.

(C) Eve peut décrypter sans problème la seconde partie M_b du message M, mais pas la première partie M_a.

(D) Eve ne peut décrypter ni la première partie M_a, ni la seconde partie M_b.

Exercice T1 (15 points) :

a) (7 points) Écrire un algorithme **algo1**(L, n) qui prenne en entrée une liste L de n nombres entiers relatifs (par exemple, $L = (+4, -12, +17, +3, -15, +34, -8, +16)$, pour $n = 8$) et dont la sortie soit la valeur maximale de la somme

$$\sum_{k=i}^{j} L(k)$$

parmi toutes les paires d'indices $i, j \in \{1, \ldots, n\}$ avec $i \leq j$ (dans l'exemple ci-dessus, la sortie doit donc être $+47$, valeur atteinte pour $i = 3$ et $j = 8$).

b) (2 points) Quelle est la complexité temporelle de votre algorithme **algo1**(L, n) ?
(utiliser la notation $\Theta(\cdot)$)

On considère maintenant l'algorithme suivant:

algo2
entrée : *liste M de n nombres entiers relatifs* sortie : *???*
Pour *i allant de 1 à $n-1$* $L(i) \longleftarrow M(i+1) - M(i)$ **Sortir : algo1**$(L, n-1)$

c) (4 points) Que représente la sortie de l'algorithme **algo2**(M, n) pour une liste M donnée ?

d) (2 points) Quelle est la complexité temporelle de l'algorithme **algo2**(M, n) ?
(utiliser la notation $\Theta(\cdot)$)

Exercice T2 (15 points) :

Considérons la séquence de 20 lettres:

DABRACADABRACADABRAC

a) (3 points) Calculer l'entropie de cette séquence, en l'exprimant tout d'abord sous la forme

$$H(X) = a + b\,\log_2(3) + c\,\log_2(5)$$

où a, b, c sont des fractions (positives ou négatives), *puis* en calculant sa valeur numérique approximative à l'aide des deux approximations $\log_2(3) \simeq 1.6$ et $\log_2(5) \simeq 2.3$.

b) (4 points) Utiliser un algorithme de compression *optimal* vu au cours pour encoder cette séquence de lettres sous la forme d'une séquence de bits : dessiner l'arbre et noter le dictionnaire obtenu.

c) (2 points) Combien de bits par lettre en moyenne votre algorithme utilise-t-il ? Votre résultat est-il cohérent avec la réponse obtenue à la question a) ? Justifier.

Pour transmettre le message ci-dessus et le protéger d'éventuelles erreurs qui pourraient survenir lors de la transmission, on vous propose maintenant trois méthodes différentes :

(I) utiliser le dictionnaire suivant: $A = 0000$, $B = 1110$, $C = 1101$, $D = 1011$ et $R = 0111$

(II) utiliser le dictionnaire suivant: $A = 000000$, $B = 111100$, $C = 001111$, $D = 101010$ et $R = 010101$

(III) prendre la séquence de bits produite à la question b), la découper en sous-séquences de 4 bits, et utiliser le codage de Hamming pour chacune de ces sous-séquences.

d) (2 points) Quel est le nombre total de bits produits par chacune de ces trois méthodes lors de l'encodage de la séquence DABRACADABRACADABRAC (identique à celle du dessus)?

On suppose maintenant qu'au plus une erreur (de type: un 0 qui devient un 1, ou le contraire) se produit sur une séquence de 8 bits consécutifs.

e) (2 points) Laquelle ou lesquelles des trois méthodes ci-dessus permettent-elles de *détecter* ces erreurs ? Justifier.

f) (2 points) Laquelle ou lesquelles des trois méthodes ci-dessus permettent-elles de *corriger* ces erreurs ? Justifier.

ICC-C Programmation

Question P1 (2 points) : Qu'est-ce que ce programme affiche ?

```c
int a = 1, b = 11, c = 111;
a += c / (b + a);
printf("%d %d %d\n", (a - b) >= (b - c), a, a == b);
```

(A)
```
0 11 1
```

(B)
```
0 10 0
```

(C)
```
1 11 1
```

(D)
```
1 10 0
```

Question P2 (2 points) : Qu'est-ce que ce programme affiche ?

```c
int m = 2;
int cnt = 0;

for (int i = 1; i <= 100; i++){
    if (i % m) cnt++;
    m++;
}

printf("%d", cnt);
```

(A)
```
49
```

(B)
```
100
```

(C)
```
50
```

(D)
```
0
```

Question P3 (2 points) : Qu'est-ce que ce programme affiche ?

```c
int n = 2048;
int cnt = 0;

do {
    n = n / 2;
    cnt++;
} while (n > 1);

printf("%d", cnt);
```

(A)
```
12
```

(B)
```
11
```

(C)
```
13
```

(D)
```
1024
```

Question P4 (2 points) : Qu'est-ce que ce programme affiche ?

```c
#include <stdio.h>
int main()
{
    int var1 = 2, var2 = -3;

    if (var2 > 0 || var1 > 0) {
        var2--;
        var1++;
    }

    if (var2 < 0 || var1 < 0) {
        var2 +=2 ;
        var1 += 2;
    }

    if (var1 && var2)
        printf("%d ", var1 * var2);
    else
        printf("%d ", -var1 * var2);

    return 0;
}
```

(A)

```
-12
```

(B)

```
4
```

(C)

```
-10
```

(D)

```
10
```

Question P5 (2 points) : Qu'est-ce que ce programme affiche ?

```c
#include <stdio.h>
#include <string.h>
int main()
{
    char line[] = "La vie est belle.";
    char c;
    int cnt = 0;

    for (int i = 0; i < strlen(line); i++){
        c = line[i];
        switch (c) {
            case 'L':
            case 'l': cnt++; break;
            default: break;
        }
    }

    printf("%d", cnt);
    return 0;
}
```

(A)

```
3
```

(B)

```
2
```

(C)

```
Ce code ne compile pas.
```

(D)

```
0
```

Question P6 (2.5 points) : Qu'est-ce que ce programme affiche ?

```c
#include <stdio.h>
#define N 10
int main() {
    int a[N] = {9, 38, 7, 31, 50, 2, 22, 27, 3, 41};
    int i = 0, cnt = 0;
    while (i < N) {
        i++;
        if (a[i] > N) continue;
        cnt += a[i];
    }
    printf("%d", cnt);
    return 0;
}
```

Question P7 (2.5 points) : Qu'est-ce que ce programme affiche ?

```c
#include <stdio.h>
#define N 10

void make_it_double(int* n){
    int res = *n * 2;
    *n = res;
}

void make_it_triple(int n) {
    int res = n * 3;
    n = res;
}

int main() {
    int a[N] = {9, 38, 7, 31, 50, 2, 22, 27, 3, 41};
    for (int i = 0; i < N; i++){
        if (i % 2) make_it_double(&a[i]);
        else make_it_triple(a[i]);
    }

    printf("%d", a[0] + a[N-1]);
    return 0;
}
```

Question P8 (2.5 points) : Combien d'étoiles ce programme affiche ?

```c
#include <stdio.h>

void print_stars(int n){
    if (n < 0) return;
    else if (n < 3) {
        printf("*");
        print_stars(n-1);
    }
    else {
        printf(" ** ");
        print_stars(n-2);
    }
}

int main() {
    print_stars(10);
    return 0;
}
```

Question P9 (2.5 points) : Qu'est-ce que ce programme affiche ?

```c
#include <stdio.h>
#define N 10

int main() {
    int a[N] = {9, 38, 7, 31, 50, 2, 22, 27, 3, 41};
    int b[N] = {0};

    for (int i = 1; i < N - 1; i++) {
        *(b + i) = *(a + i + 1) + *(a + i - 1) - *(a + i);
    }
    printf("%d", *(b + N/2));
    return 0;
}
```

Question P10 (2.5 points) : Qu'est-ce que ce programme affiche ?

```c
#include <stdio.h>

int main() {
    int x[2] = {1, 2};
    int y[2] = {3, 4};
    int z[2] = {5, 6};
    int *p1, *p2;

    p1 = x;
    p2 = z + 1;

    *(p1 + 1) += *p2;
    p2 -= 1;

    p1 = y;
    *(p1 + 1) = *p2;

    *p2 = *p1;

    printf("%d", x[1] + y[1] + z[1]);

    return 0;
}
```

Question P11 (2.5 points) : Qu'est-ce que ce programme affiche ?

```c
#include <stdio.h>
#include <string.h>

int main() {
    char s[100] = "Bond. James Bond.";
    char t[100] = {'\0'};
    char* key = "Bond.";
    int cnt = 0;
    char *cp = strstr(s, key);

    while (cp != NULL) {
        strcat(t, key);
        cnt++;
        cp = strstr(s + cnt * strlen(key), key);
    }

    printf("%d", cnt);
    return 0;
}
```

Question P12 (5 points) : Écrivez la fonction `count_digits` qui attend deux arguments:

- un entier positif `n` et
- un tableau `d` de dix éléments de type `int`, tous initialisés à zéro.

Cette fonction doit compter le nombre de fois qu'un chiffre (0, 1, 2, ..., 9) apparaît dans le nombre `n` et remplir le tableau `d` en conséquence. L'élément du tableau `d` à l'indice i correspond au nombre d'occurrences du chiffre i. Pour un `n` négatif ou égal à zéro, la fonction ne doit apporter aucune modification au tableau `d`.

Exemples:

(1) `count_digits(155027,d)` aura comme effet :

indice i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
d[i]	1	1	1	0	0	2	0	1	0	0

(2) `count_digits(8285260,d)` aura comme effet :

indice i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
d[i]	1	0	2	0	0	1	1	0	2	0

(3) `count_digits(1575559,d)` aura comme effet :

indice i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
d[i]	0	1	0	0	0	4	0	1	0	1

Question P13 (5 points) : Écrivez une fonction `pearson_corr` qui attend deux arguments:

- `x`, un tableau d'éléments de type `double`,
- `y`, un tableau d'éléments de type `double` et
- `n`, le nombre d'éléments dans chacun des tableaux `x` et `y`.

Votre fonction doit retourner le coefficient de corrélation de Pearson r, qui se calcule à l'aide de formule suivante :

$$r = \frac{\sum_{i=1}^{n} \Big((x_i - \bar{x}) \cdot (y_i - \bar{y}) \Big)}{\sqrt{\Big(\sum_{i=1}^{n} (x_i - \bar{x})^2 \Big) \cdot \Big(\sum_{i=1}^{n} (y_i - \bar{y})^2 \Big)}} \tag{1}$$

où $\bar{x}$ et $\bar{y}$ sont les moyennes arithmétiques du tableau `x` et `y`, respectivement:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \bar{y} = \frac{\sum_{i=1}^{n} y_i}{n} \tag{2}$$