

Nom et prénom : Spinelli Isaia

## Travail écrit no. 2

(Durée : 2 périodes)

### Directives :

- ECRIVEZ VOS REPONSES DIRECTEMENT SUR LA DONNEE
- Ne pas dégrafer le document
- Vous pouvez écrire au crayon
- Seule documentation autorisée : la Quick Reference Card C (non annotée !)

5.75

### Question 1 (7.5 points)

0.75

a) (2.5 pts)

Pour chacune des suites d'instructions ci-dessous, indiquer dans la colonne de droite du tableau la valeur affichée par le printf.

non

Suite d'instructions	Valeur affichée
<pre>#define double(d) d + d int n = 5; printf("%g\n", double((double)(n+2)/5));</pre> <p>3 → 011 101 110 001 010 → -2</p>	208 0.75
<pre>#define ABS(a) (a) &lt; 0 ? -a : a int n = -3; printf("%d\n", ABS(n^n));</pre> <p>(-3) &lt; 0 ? -(-3) : (-3) → -2</p>	-2

(x) = priorité

2.5

b) (2.5 pts)

Que va afficher le programme suivant ?

0011 = 3  
1101 = -3

^ = XOR (l'un ou l'autre mais pas les 2)

```
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>

uint16_t f(uint16_t i, uint8_t j) {
    for (uint8_t k = 0; k < j; ++k)
        i = (uint16_t) ((i >> 1) | (i & 0x0001 ? 0x8000 : 0));
    return i;
}

int main(void) {
    printf("%#X\n", f(0x1357, 9));
    return EXIT_SUCCESS;
}
```

Votre réponse :

0xAB89 ✓

2.5

c) (2.5 pts)

Que va afficher le programme suivant si `&t[0][0]` est supposé valoir `0x61fe20` ?

```
#include <inttypes.h>
#include <stdio.h>
#include <stdlib.h>

#define PRINT_ADDRESS(ADR) printf("0x%" PRIxPTR "\n", (intptr_t) (ADR))

int main(void) {
    int16_t t[][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    printf("1) "); PRINT_ADDRESS(t[2] + (&t[1][2] - t[1]));
    printf("2) "); PRINT_ADDRESS(*(t + 2));
    return EXIT_SUCCESS;
}
```

*Handwritten annotations:*  
- Above the array definition: 22 24 26 28 30 with arrows pointing to the first three elements of each row.  
- Above the first printf: 2C 2E 30 with arrows pointing to the elements 4, 5, and 6.  
- Below the second printf: 2, 2+4 with an arrow pointing to the address of the third row.

Vos réponses :

- 1) `0x61fe30` ✓
- 2) `0x61fe2C` ✓

6.75

## Question 2 (7.5 points)

Que va afficher le programme suivant, après que l'utilisateur ait saisi la suite de caractères "(024) 1234567 - RRH " ?

AA AB 5 3

### IMPORTANT

- Dans vos réponses, désignez par la **lettre b** la présence d'un espace blanc.
- Seules les réponses 100% correctes seront comptabilisées.

```
#include <stdio.h>
#include <stdlib.h>

#define TAILLE_MAX 50

int main(void) {

    int n = 171;
    printf("1) |%2d|\n", n);
    printf("2) |%-3X|\n", n);
    printf("3) |%#5o|\n", n);

    double x = 0.2468;
    printf("4) |%f|\n", x);
    printf("5) |%+4.1f|\n", x);
    printf("6) |%.2e|\n", x);
    printf("7) |%g|\n", x);

    printf("8) |%G|\n", .5E6);
    printf("9) |%5.2g|\n", 1.2345);

    char chaine[TAILLE_MAX + 1] = "";
    printf("Entrez une chaine de caracteres (%u caract max) > ", TAILLE_MAX);
    scanf("%*[(0123456789)]", chaine);
    printf("10) |%s|\n", chaine);

    return EXIT_SUCCESS;
}
```

### Vos réponses :

- 1) 1) b | 171 | ✓
- 2) 2) b | ABb | ✓
- 3) 3) b | 0253 | ✓
- 4) 4) b | 0.246800 | ✓
- 5) 5) b | +0.2 | ✓
- 6) 6) b | 2.47e-001 | ✓
- 7) 7) b | 0.2468 | ✓
- 8) 8) b | 500000 | ✓
- 9) 9) b | 1.23 | ✓
- 10) 10) b | | ✓

128 + 90 + 3

7.5

### Question 3 (7.5 points)

3.75

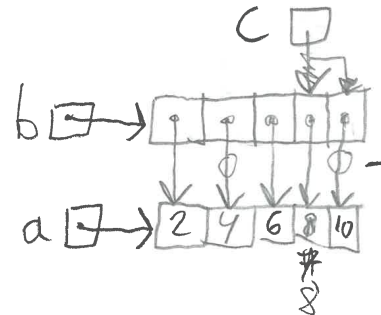
- a) En supposant les <?> du code ci-dessous remplacés par le code de format adéquat, indiquer ce que va afficher le programme suivant :

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    int a[] = {1, 4, 6, 8, 10};
    int* b[] = {a, a+1, a+2, a+3, a+4};
    int** c = &b[0];

    printf("1) <?>\n", **b+1);
    printf("2) <?>\n", b[0][1]);
    printf("3) <?>\n", --**c);
    printf("4) <?>\n", ++**c++);
    printf("5) <?>\n", *c - *(b+1));

    return EXIT_SUCCESS;
}
```



#### Vos réponses :

- 1) 3 ✓
- 2) 4 ✓
- 3) 7 ✓
- 4) 8 ✓
- 5) 3 ✓

3.75

- b) En supposant les <?> du code ci-dessous remplacés par le code de format adéquat, indiquer ce que va afficher le programme suivant :

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {

    const char* a[] = {"ABC", "DEFG", "HIJ"};
    const char** b[] = {a, a+1, a+2};
    const char*** c = &b[2];

    printf("1) <?>\n", **a);
    printf("2) <?>\n", *a[1]);
    printf("3) <?>\n", *b[2]);
    printf("4) <?>\n", *--*(b+1) + 1);
    printf("5) <?>\n", *(c[-1][1] + 2));

    return EXIT_SUCCESS;
}
```

**Vos réponses :**

- 1) 'A' ✓
- 2) 'D' ✓
- 3) 'HIJ' ✓
- 4) 'BC' ✓
- 5) 'F' ✓

3

### Question 4 (6 points)

1) Traduire en français les déclarations C suivantes :

a) `int* (*a(char* const))[5];`

b) `int (*(*b[5])(int))[10];`

2) Ecrire les déclarations C correspondant aux énoncés suivants :

*Attention! Des points seront déduits en cas de surparenthésage*

a) p est un pointeur sur une fonction prenant en paramètre un pointeur sur un tableau de 5 int et livrant un pointeur sur un double constant

b) f est une fonction sans paramètre livrant un pointeur sur un tableau de 5 pointeurs constants sur double

Vos réponses :

1) a) a est un pointeur sur une fonction prenant en paramètre un pointeur constant sur un char et livrant un pointeur sur un tableau de 5 pointeurs sur int

b) b est un tableau de 5 pointeurs pointant sur une fonction prenant en paramètre un int et livrant un pointeur sur un tableau de 10 int. ✓

2) a) ~~const double\* (\*p)(int(\*)[5]);~~

b) ~~double const\* f(void)[5]~~

a) `const double* (*p)(int(*)[5]);` ✓

b) `double* const (*f(void))[5];` ! Essentiel! Sinon faux

3.75

### Question 5 (7.5 points)

La fonction *différence* proposée ci-dessous comporte diverses erreurs.

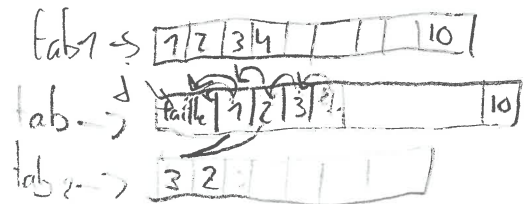
Indiquer le no des lignes problématiques et proposer dans chaque cas un correctif complet écrit en C.

#### IMPORTANT

- Une réponse avec no de ligne problématique correct mais sans proposition de correctif sera considérée comme fausse. Idem si le correctif proposé est faux.
- Des points seront décomptés si une réponse, au lieu de corriger une erreur, en ajoute une supplémentaire.
- Aucune ligne de code ne doit être ajoutée, supprimée ou déplacée.

```
// Retourne un tableau comprenant tous les éléments de tab1 qui ne figurent pas
// dans tab2.
// Retourne NULL si la mémoire est insuffisante pour créer le tableau solution.
// Remarques :
// 1) La fonction présuppose que ni tab1, ni tab2 ne sont NULL.
// Le cas échéant, la fonction provoque une assert error à l'exécution
// 2) Le premier élément du tableau solution contient le nombre d'éléments
// contenus dans ce dernier.
```

```
1 int* difference(const int* tab1, size_t taille1,
2               const int* tab2, size_t taille2) {
3     assert(tab1 != NULL && tab2 != NULL);
4     int* tab = (int*) malloc(taille1 + 1, sizeof(int));
5     if (!tab) {
6         assert(taille1 <= INT_MAX);
7         *tab = (int) taille1;
8     }
9     memcpy(tab + 1, tab1, taille1);
10    for (size_t i = 0; i < taille2; ++i) {
11        size_t j = 0;
12        for (size_t k = 1; k <= (size_t) *tab; ++k)
13            if (tab[k] != tab2[i])
14                tab[j++] = tab[k];
15    }
16    tab = realloc(tab, (size_t) (tab[0] + 1) * sizeof(int));
17    return tab;
18 }
19 }
```



tab = tout les elements de tab1 sans ceux qui apparaissent dans tab2.

Votre réponse :

✓ ligne 1: int\* difference ... non int\*\*

✓ ligne 5: if (tab)

✓ ligne 8: memcpy(tab + 1, tab1, taille1 \* sizeof(int));

✓ ligne 10: size\_t j = 0; → ligne 14: j-1 ou ligne 13: tab[j]

✓ ligne 16: ~~tab~~ = realloc(tab, (size\_t) (\*tab \* sizeof(int)));

↑ unique faute  
ligne 11: k <= (size\_t) \*tab

Votre réponse (suite) :

5.25

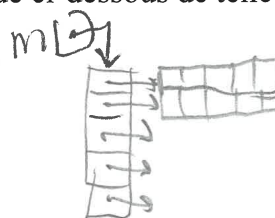
### Question 6 (8 points)

Compléter les 4 parties notées **<à compléter>** du code ci-dessous de telle sorte que celui-ci affiche à l'exécution :

```

0      0 4
1      X...X
2      1.X.X. 3
3      2..X..
4      1.X.X. 3
5      0X...X 4

```



#### IMPORTANT

- **<à compléter 3>** doit être implémenté de la manière **la plus efficace possible**
- Dans le code ci-dessous, les `#include` ont volontairement été omis.

```

void initialiser(char* matrice, size_t n,
                char surDiagonales, char horsDiagonales);
void afficher(const char* matrice, size_t n);

int main(void) {
    #define TAILLE 5
    char matrice[TAILLE][TAILLE];

    initialiser(<à compléter 1>, TAILLE, 'X', '.');
    afficher(<à compléter 2>, TAILLE);

    return EXIT_SUCCESS;
}

void initialiser(char* matrice, size_t n,
                char surDiagonales, char horsDiagonales) {
    <à compléter 3>
}

void afficher(const char* matrice, size_t n) {
    for (size_t i = 0; i < n; ++i) {
        for (size_t j = 0; j < n; ++j)
            printf("%c", <à compléter 4>);
        printf("\n");
    }
    printf("\n");
}

```



Votre réponse :

(char\*) matrice

à compléter 1:

\*matrice ✓

à compléter 2:

\*matrice ✓

tout mettre à '\0' (très efficace)

à compléter 3:

for (size\_t i = 0; i < n; ++i) {

memset(matrice, horstDiagonal, n\*n);

for (size\_t k = 0; k < n; ++k) {  
if (k == i || k == (n-1-i))

for (size\_t i = 0; i < n; ++i)

matrice[(n-1)\*i] = matrice[(n-1)\*(i+1)] =

surDiagonales;

\*matrice + (i\*n) + k = surDiagonales;

else  
\*matrice + (i\*n) + k = horstDiagonales;

}

Complexité de  $O(n)$ ;

}

⚠ Algo peu efficace!

à compléter 4:

\*matrice + (i\*n) + j ✓

