5.8 Bien!

Nom et prénom : Friedli Jonathan

Travail écrit no. 1

(Durée : 2 périodes)

Consignes :
- **ECRIVEZ VOS REPONSES DIRECTEMENT SUR CE DOCUMENT**
- Seules les réponses figurant sur ce document seront corrigées
- Vous pouvez écrire au crayon papier
- Ne pas dégrafer ce document
- Un résumé C figure à la fin de ce document

4

Question 1 (6 points)

2 a) (2 pts)
Pour chacune des suites d'instructions ci-dessous, indiquer dans la colonne de droite du tableau la valeur affichée par le printf.

Suite d'instructions	Valeur affichée

`#define double(d) d + d`
`int n = 1;`
`printf("%g\n", double((double)(n+2)/5));`
	1.2 ✓

`#define ABS(a) ((a < 0) ? (-a) : (a))`
`int n = 5;`
`printf("%d\n", ABS(n^n));`
	-2 ✓

2 b) (2 pts)
Que va afficher le programme suivant ?

```
#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(0x1234, 5));
   return EXIT_SUCCESS;
}
```

Votre réponse : 0xa091 ✓

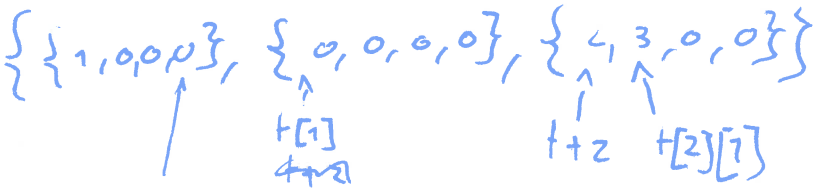
0 c) (2 pts)
Que va afficher le programme suivant si &t[0][0] est supposé valoir 0x61fdd0 ?

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

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

int main(void) {
   int32_t t[][4] = {{1}, {}, {2, 3}};
   PRINT_ADDRESS(t[1] - (*(t + 2) - &t[2][1]));
   return EXIT_SUCCESS;
}
```

Votre réponse :



$0x61fdd0 + 3 \cdot \frac{3 \cdot 8^2}{8}$

=> 0x61fddc ox61fde4

4.8

Question 2 (6 points)

Que va afficher le programme suivant ?

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>

int main(void) {

   printf("1) |%2d|\n", 173);
   printf("2) |%-3X|\n", 173);
   printf("3) |%#.5o|\n", 173);

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

   printf("8) |%#G|\n", .5E5);
   printf("9) |%05.3g|\n", 12.345);

   // On suppose que l'utilisateur entre la chaîne " (012) 3456789 - SOS "
   #define TAILLE_MAX 50
   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) b|173|✓
2) b|AD|✓
3) b|0255| b|00255|
4) b|0.246800|✓
5) 5) b|+0.2|✓
6) b|2.47e-001|✓
7) b|0.2468|✓
8) b|50000| b|50000.0|
9) b|012.3|✓
10) b|012|✓

4.8

Question 3 (6 points)

3

a) (3 pts)
Soient les déclarations suivantes :

```
int a[] = {1, 3, 5, 7, 9};
int* b[] = {a, a+1, a+2, a+3, a+4};
int** c = &b[3];
```

Quelle valeur fournit chacune des expressions ci-dessous ?
(Conseil : Aidez-vous d'un petit dessin)

1) `**b+1`
2) `c[-1][-1]`
3) `--**c`
4) `++**c++`
5) `*c + 1 - *(b+1)`

IMPORTANT

- Les 5 expressions sont supposées être évaluées les unes après les autres, en commençant par la no 1). Le résultat d'une certaine expression peut donc dépendre de ce qui s'est produit lors de l'évaluation des expressions précédentes.

Vos réponses :

1) 2 ✓
2) 3 ✓
3) 6 ✓
4) 7 ✓
5) 4 ✓

1.8

b) (3 pts)
Soient les déclarations suivantes :

```
const char* a[] = {"Federer", "Nadal", "Djokovic"};
const char** b[] = {a, a+1, a+2};
const char*** c = b + 2;
```

Quelle valeur fournit chacune des expressions ci-dessous ?
(Conseil : Aidez-vous d'un petit dessin)

1) `**a`
2) `*a[1]`
3) `*b[2]`
4) `*(*--*(b+1) + 1) + 1`
5) `*++c[-1][1]`

IMPORTANT

- Les 5 expressions sont supposées être effectuées les unes après les autres, en commençant par la no 1). Le résultat d'une certaine expression peut donc dépendre de ce qui s'est produit lors de l'évaluation des expressions précédentes.

Consigne

- Pour les caractères, donner le résultat entre simples guillemets.
  Exemple : 'a'.
- Pour les chaînes de caractères, donner le résultat entre doubles guillemets.
  Exemple : "ABC".

Vos réponses :

1) 'F' ✓
2) 'N' ✓
3) "Djokovic" ✓
4) ~~'a'~~ 102 (code ascii de (e+1))
5) ~~'D'~~ 'a'

6

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 :

3 / 1.5

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

1.5

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

3 / 1.5

2) a) `const double * (*p)(int(*)[5]);` ✓

1.5

b) `double *const (*f(void))[5];` ✓

1.5

6

Question 5 (6 points)

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

```
X...X
.X.X.
..X..
.X.X.
X...X
```

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) {
   assert(matrice != NULL);
   assert(n > 0);
   <à compléter 3>
}

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

Votre réponse :

<à compléter 1> `&matrice[0][0];`

<à compléter 2> `&matrice[0][0];` ✓

<à compléter 3>

~~memset(matrice~~

~~memset(&~~

```
memset(matrice, horsDiagonales, n*n); ✓
for (size_t i=0; i<n; i++){
    matrice[(n+1)*i] = sunDiagonales;
    matrice[(n-1)*(i+1)] = sunDiagonales;
}
```

✓ Bien !

<à compléter 4> `matrice[n*i + j]` ✓

Question 6 (6 points)

Ecrire, de la manière la plus efficace possible, une fonction C, sans valeur de retour, permettant d'inverser les éléments d'un tableau à 1 dimension et ce, quelle que soit la taille du tableau et quel que soit le type des éléments du tableau.

Exemples

- Si, en entrée, tab = [1, 2, 3] (**tableau de int**), alors la fonction doit produire en sortie : tab = [3, 2, 1]
- Si, en entrée, tab = [1.1, 2.2, 3.3, 4.4] (**tableau de double**), alors la fonction doit produire en sortie : tab = [4.4, 3.3, 2.2, 1.1]

Votre réponse :

```c
void inverser(void* debut, void* fin, size_t size) {
    assert(debut != NULL);
    assert(fin != NULL);
    assert(size > 0);

    char* tampon = (char*) calloc(size, 1); // calloc(size, sizeof(char))
                                            // calloc(size, sizeof(*tampon))
    assert(tampon != NULL);

    char *ptr1 = (char*) debut,
         *ptr2 = (char*) fin;

    while (ptr1 < ptr2) {
        memcpy(tampon, ptr1, size);
        memcpy(ptr1, ptr2, size);
        memcpy(ptr2, tampon, size);
        ptr1 += size;
        ptr2 -= size;
    }

    free(tampon);
}
```

On demandait d'écrire une fonction, pas une macro !