

Nom et prénom :

Spinelli Isaïa

Travail écrit no. 3

(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 !)

Question 1 (6 points)

Pour chacune des instructions ou suites d'instructions ci-dessous, indiquer dans la colonne de droite du tableau si celle-ci est *juste* ou *fausse*.

IMPORTANT

- Il n'est PAS demandé de justifier vos réponses ou de proposer un quelconque correctif
- Barème appliqué : +0.5 point pour une réponse correcte; 0 point en cas d'absence de réponse; -0.5 point pour une réponse incorrecte

Instructions	Juste ou faux ?
<code>char chaine[3] = "ABC";</code>	faux ✓
<code>char chaine[10];</code> <code>chaine = "ABC";</code> ← pte constant	juste Faux
<code>const char* chaine1;</code> <code>char chaine2[] = "ABC";</code> <code>chaine1 = chaine2;</code>	juste ✓
<code>typedef char string[];</code>	faux juste
<code>struct S {</code> <code>int n = 1;</code> <code>double x = 2.5;</code> <code>}</code> <code>S;</code>	faux ✓
<code>struct S {</code> <code>const int N;</code> <code>double x;</code> <code>};</code> <code>struct S s = {1, 2.5};</code> init à la création	Juste

<pre>struct S { int n; double x; }; struct S s; s = (S){1, 2.5};</pre> <p><i>struct</i></p>	<p>juste Faux</p>
<pre>struct S { int n; double x; }; struct S s1 = {1, 2.5}; struct S s2 = s1;</pre>	<p>juste ✓</p>
<pre>struct S { int n; double x; }; struct S s1 = {1, 2.5}; struct S s2 = {1, 2.5}; printf("s1 == s2 ? %s\n", (s1 == s2 ? "oui" : "non"));</pre>	<p>juste Faux</p>
<pre>struct S { enum {A, B, C = 0} e; }; printf("%d\n", A);</pre>	<p>faux juste</p>
<pre>struct S1 { struct S2* a; }; struct S2 { struct S1* a; };</pre> <p><i>il connaît la taille d'une adresse</i></p>	<p>faux juste</p>
<pre>union U { int n; double x; }; union U u = {1.5}; printf("u = %f\n", u.x);</pre>	<p>faux ✓</p>

~~Elem~~
typedef struct {
 Elem*
 Elem
} Elem;

6

Question 2 (6 points)

a) Soient les déclarations suivantes :

3.75

```
const char* t[] = {"matrice", ":", "tableau", "a", "2", "entrees"};  
const char** pt[] = {t+6, t, t+3, t+1, t+2, t+4, t+5};
```

Quelle valeur fournit chacune des expressions ci-dessous ?

(Conseil : Aidez-vous d'un petit dessin)

- 1) $*pt - pt[3]$
- 2) $*pt[2]$
- 3) $**(*pt-3)+2$
- 4) $t[0]+2$
- 5) $*pt[++pt[2]-*(pt+4)-1][2]-1$

IMPORTANT ↴

- Pour les caractères, donner le résultat entre apostrophes. Ex: 'a'.
- Pour les chaînes de caractères, donner le résultat entre guillemets. Ex : "ABC".

Vos réponses :

- 1) 5 ✓
- 2) "a" ✓
- 3) 99 ('x') ✓
- 4) "trice" ✓
- 5) 115 ('s') ✓

2.25 b) Que va afficher, à l'exécution, le programme C suivant ?

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

int main(void) {
    int t[] = {3, 6, 9};
    int* p;
    for (int i = 0; i < 3; ++i) {
        p = &t[i];
        while (p < &t[3])
            *p++ += *(t+i);
        for (int i = 0; i < 3; ++i)
            printf("%d ", t[i]);
        printf("\n");
    }
    return EXIT_SUCCESS;
}
```

Votre réponse :

6 12 15 ✓
6 24 39 ✓
6 24 78 ✓

1.75 **Question 3 (6 points)**

La fonction *matriceTriangulaireInferieure* ci-dessous a pour objectif de livrer en valeur de retour la "matrice triangulaire inférieure gauche" correspondant à la matrice carrée de taille $n \times n$ passée en paramètre et *NULL* si elle ne peut mener à bien sa tâche¹.

¹ Les cas où la matrice passée en paramètre vaudrait *NULL* et/ou n vaudrait 0 ne sont pas considérés ici.

Exemple

Pour la matrice carrée $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$, la fonction doit renvoyer la matrice irrégulière² $\begin{bmatrix} 1 \\ 4 & 5 \\ 7 & 8 & 9 \end{bmatrix}$

² La première ligne contient 1 seul élément (ici 1), la seconde ligne 2 éléments (ici 4 et 5), etc.

Compléter les 4 parties notées **<à compléter>** du code ci-dessous de telle sorte qu'il produise le résultat escompté.

$n = 3$

```
int** matriceTriangulaireInferieure(const int* matrice, size_t n) {
    int** resultat = <à compléter 1>;
    if (resultat) {
        for (size_t i = 0; i < n; ++i) {
            resultat[i] = <à compléter 2>;
            if (resultat[i]) {
                <à compléter 3>;
            } else {
                <à compléter 4>;
            }
        }
    }
    return resultat;
}
```

IMPORTANT

- **<à compléter 3>** doit être implémenté au moyen d'une seule instruction (pas de boucle !)

Vos réponses :

<à compléter>	Votre proposition
1 1	$(int^{**}) \text{ calloc}(n, \text{sizeof}(int^{*}))$ ✓
2 0	matrice + (i * n) $(int^{*}) \text{ calloc}(i+1, \text{sizeof}(int));$
3 0.25	<u>memcpy</u> (* resultat[i], * (matrice + (i * n)), (i+1) * sizeof(int))
4 0.5	Manque l'essentiel, cf corrigé en classe $\text{resultat} = \text{NULL};$ $\text{for}(\text{size_t } j=0; j < i; ++j) \{$ $\quad \text{free}(\text{resultat}[j]); \}$ $\text{free}(\text{resultat});$

memcpy préférable car ici pas de "chevauchement" en mémoire

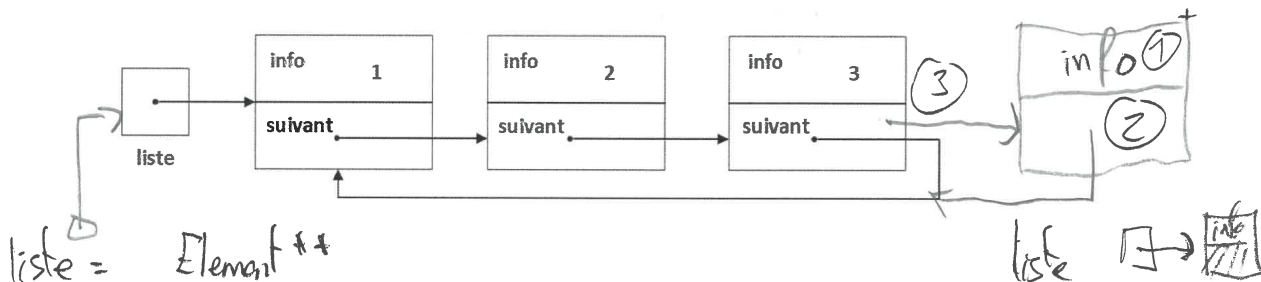
6

Question 4 (9 points)

On suppose disposer des déclarations (non modifiables) suivantes, censées permettre la modélisation d'une **liste dynamique simplement chaînée circulaire** (voir figure ci-après) :

```
typedef int Info;  
  
typedef struct element {  
    Info info;  
    struct element* suivant;  
} Element;  
  
typedef Element* Liste;
```

Exemple de liste simplement chaînée circulaire avec 3 éléments



Proposer une implémentation de la fonction dont le prototype est le suivant :

```
// Insère un nouvel élément (contenant info) en queue de liste.  
// Renvoie true si l'insertion s'est déroulée avec succès et false  
// dans le cas contraire.
```

```
bool insererEnQueue(Liste* liste, const Info* info);
```

IMPORTANT

- Les cas où l'utilisateur de la fonction passerait *NULL* en paramètre effectif pour *liste* ou pour *info* ne sont pas à traiter ici.
- Commenter succinctement votre code de manière à en faciliter la compréhension

Votre réponse :

```
bool insertInQueue(Liste* liste, const Info* info)
{
    //Crée un new element
    Element* elem = (Element*) malloc(sizeof(Element)); ✓
    if((*liste)→suivant) // si c'est pas vide
    {
        *elem = (Element) { *info, *liste };
        while((*liste)→suivant != *liste) // cherche l'ancien dernier element
            ++(*liste);
        *liste→suivant = elem; // accorde l'ancien dernier element au nouveau
    }
    else // si c'est vide
    {
        *elem = (Element) { *info, elem }; // suivant est lui même
        *liste = elem;
    }
    return true;
}

if(!elem)
    return false; ✓
```

Idee juste mais faut un itérateur! sinon faux

pas à traiter (cf donnée)

10

Question 5 (11 points)

1) (9 pts)

Ecrire, de la manière la plus propre et la plus modulaire / évolutive possible, toutes les déclarations de constantes et de types (**et rien d'autre !**) permettant de modéliser des vaisseaux spatiaux conformément au cahier des charges suivant :

- Un vaisseau spatial est soit un vaisseau de combat, soit un vaisseau d'exploration.
- Tout vaisseau spatial a un nom (de longueur quelconque) et possède un équipage¹.
¹ Tout vaisseau comporte un équipage... mais un équipage peut éventuellement être vide
- Un équipage comprend de 0 à 5 membres au maximum.
- Un membre d'équipage se caractérise, pour l'heure, uniquement par son nom (de longueur quelconque).
- Si le vaisseau spatial est un vaisseau de combat, on souhaite enregistrer, en plus de son nom et de son équipage, son poids (nombre **entier** exprimé en kg) et s'il est équipé ou non de canons laser.
- Si le vaisseau spatial est un vaisseau d'exploration, on souhaite enregistrer, en plus de son nom et de son équipage, son rayon d'action (nombre **réel** exprimé en milliards de km).

2) (2 pts)

En supposant le point 1) résolu, déclarer les 2 vaisseaux spatiaux suivants :

- "Starfighter" : un vaisseau de combat de 2500 kg, équipé de canons laser et ayant pour équipage Joe et Jack
- "X-Wing" : un vaisseau d'exploration sans équipage dont le rayon d'action est de 63.2 milliards de km

IMPORTANT

Les déclarations des 2 vaisseaux spatiaux doivent être implémentées chacune à l'aide d'**une seule instruction écrite de la manière la plus courte possible**¹.

¹ La présence de toutes les paires d'accolades est toutefois requise.

1)

Votre réponse :

```
#include <stdbool.h>

#define MAX_MEMBRES 5 ✓

typedef enum { COMBAT, EXPLORATION } TypeVaisseau;

typedef struct {
    const char* nom;
} Membre; ✓

typedef struct {
    size_t nbMembre;
    Membre equipe[MAX_MEMBRES];
} Equipage; ✓

typedef struct {
    unsigned poids; // kg
    bool laser;
} Combat; ✓

typedef struct {
    double rayon; // milliards de km
} Exploration; ✓

typedef union {
    Combat combat;
    Exploration exploration;
} GenreVaisseau; ✓
```

Votre réponse (suite) :

```
typedef struct {  
    const char* nom;  
    TypeVaisseau type;  
    Equipage equipe;  
    GenreVaisseau vaisseau;  
} Vaisseau Spatial;  
tail !
```

2).

Vaisseau Spatial $V_1 = \{ \text{"Starfighter"}, \text{COMBAT}, \{ 2, \{ \{ \text{"Joe"} \}, \{ \text{"Jack"} \} \} \}$ 2500, true };

Vaisseau Spatial $V_2 = \{ \text{"X-Wing"}, \text{EXPLORATION}, \{ 0, \{ \} \}, \{ \text{exploration} = \{ 6302 \} \} \}$;

