Université Internationale de Casablanca LAUREATE INTERNATIONAL UNIVERSITIES Nous innovons pour votre réussite !	**Ecole d'Ingénierie** **Filières :** CPI **Classe :** 2ème année	**Cours : Programmation Structurée 2** **Professeur** : MOUJAHID Abdallah **Date** : 15/01/2016

Examen Final

Type : A

Durée : 2 heures

Etudiant :	
Note :	

Notes Importantes :

- *Aucun document autorisé. Sont interdits tous les calculatrices, les téléphones, ainsi que tout autre outil de calcul et/ou de communication.*
- *Vous devez aussi remettre à votre professeur cet imprimé, portant votre nom, (Un étudiant qui n'a pas remis l'imprimé n'aura pas de note)*
- *TOUTE sortie est définitive !*
- *TOUTE tentative de fraude sera sanctionnée (Avertissement pour la première fois, -5 la deuxième fois, 0/20 la troisième fois)*

Partie I : QCM *(10 points)*

Pour chaque question, il y a exactement une bonne réponse. Veuillez répondre dans la grille en mettant une croix dans la case correspondante.
+0,5 pour une bonne réponse, 0 pour absence de réponse, -0,25 pour une mauvaise réponse.

1. **Laquelle de ces valeurs est la plus grande ?**
   a. sizeof(int *)
   b. sizeof(char *)
   c. sizeof(double *)
   d. les 3 ont la même taille.

2. **Pour récupérer au clavier un entier et deux caractères à placer dans une variable i et les deux cases t[0] et t[1] d'un tableau de deux caractères, on peut écrire :**
   a. scanf ("%d%c%c", &i, t, t + 1)
   b. scanf ("%d%c%c", &i, *t, *(t + 1))
   c. scanf ("%d%c%c", i, t[0], t[1])
   d. scanf ("%d%c%c", &i, t)

3. **Combien d'int peut-on stocker au maximum dans un tableau tab alloué avec la commande : int * tab = (int *) malloc(1000) ; ?**
   a. 1000
   b. 1000/sizeof(int)
   c. 1000×sizeof(int)
   d. on ne peut pas savoir

4. **Quel sera le résultat du code suivant :**

```
#include <stdio.h>
int main() {
        int a, b;
        int *ptr1, *ptr2;
        a = 5;
        b = a;
        ptr1 = &a;
        ptr2 = ptr1;
        b = (*ptr2)++;
        printf("a = %d, b = %d, *ptr1 = %d, *ptr2 = %d\n", a, b, *ptr1, *ptr2);
        return 0;
}
```

   a. ne compile pas
   b. provoque une erreur à l'exécution.

c. affiche a = 6, b = 5, *ptr1 = 6, *ptr2 = 6
d. affiche a = 5, b = 5, *ptr1 = 5, *ptr2 = 5
e. affiche a = 6, b = 6, *ptr1 = 6, *ptr2 = 6

5. Qu'affiche le programme suivant :

```
int main() {
int x = 2;
switch (x) {
        case 1: x = 4;
        case 2: x = 5;
        case 3: x = 6;
        default: {}
}
printf("%d\n", x);
return 0; }
```

a. 2
b. 5
c. 6
d. rien car il n'est pas correct

6. Si x a été alloué avec : *int *x = malloc(42 * sizeof(int));* alors, pour libérer la mémoire pointée par x, on peut :

a. Atteindre la fin de la portée de la variable x, la libération sera automatique.
b. Exécuter for (int i = 0; i < 42; i++) free(x + i);
c. Exécuter free(x);
d. Exécuter free(x, 42 * sizeof(int));
e. Rien, la zone pointée par x sera désalloué toute seule quand elle ne sera plus utilisée.

7. Quel sera le résultat du code suivant :

```
#include <stdio.h>
#define TAB_LENGTH 3

int main() {
        int tab[TAB_LENGTH];
        int j;
        for (j = 0; j < TAB_LENGTH; j++)
                tab[j] = 5;
```

```
        j = 0;
        printf("[");
        while (j < TAB_LENGTH)
                printf(" %d ", tab[j]); j++;
        printf("]");
        return 0;
}
```

a. ne compile pas
b. provoque une erreur fatale à l'exécution
c. boucle
d. affiche [5 5 5]
e. affiche autre chose

8. On cherche à libérer entièrement la mémoire occupée par une liste chaînée. Laquelle de ces fonctions effectue cette opération correctement ?

a.
```
void free_list (cell* L) {
  if (L != NULL) {
    free_list(L->next) ;
    free(L) ;
  }
}
```

b.
```
void free_list (cell* L) {
  if (L != NULL) {
    free_list(L) ;
    free(L->next) ;
  }
}
```

c.
```
void free_list (cell* L) {
  if (L != NULL) {
    free(L) ;
    free_list(L->next) ;
  }
}
```

d.
```
void free_list (cell* L) {
  if (L != NULL) {
    free(L->next) ;
    free_list(L) ;
  }
}
```

9. Quel sera le résultat du code suivant :

```
#include <stdio.h>
#define TAB_LENGTH 3
int main() {
        int tab[TAB_LENGTH];
        int j = 0;
        int *ptr = &tab[3];
        for(; j < TAB_LENGTH; j++)
                tab[j] = 5;
        *(tab + 1) = 3;
        *(ptr - 1) = 3;
        printf("[ %d %d %d ]", tab[0], tab[1], tab[2]);
        return 0;
```

```
}
```

a. ne compile pas
b. provoque une erreur fatale à l'exécution.
c. boucle
d. affiche [3 5 3]
e. affiche [3 3 5]
f. affiche [5 3 3]
g. affiche [5 3 5]

10. Le message suivant: "warning: 'return' with a value, in function returning void" est affiché lors de la compilation pour la fonction :

a. void f() { return 1; }
b. int g() {}
c. char h() { return 7.0; }
d. aucune de ces fonctions

11. Comment affiche-t-on le champ 'nom' de la variable 'test' dans l'exemple suivant:
struct { char nom; } test;*

a. printf("%s", test.nom);
b. printf("%s", test->nom);
c. printf("%s", *test.nom);
d. printf("%s", &test->nom);

12. Indiquer quel est l'appel correct pour la signature suivante :

*Void sp (int *i ,double j);*

int a ;

double b ;

a. sp(a, b);
b. sp(a, &b);
c. sp(&a, b);
d. sp(*a, b);

13. Par laquelle des propositions suivantes faut-il remplacer la ligne ..... dans le code ci-dessus pour être certain qu'il s'exécute correctement ?

```
1  int main(int argc, char* argv[]) {
2    int i;
3    int* tab;
4    ........
5    tab[0] = 2;
6    for (i=1; i<=5; i++) {
7      tab[i] = tab[i-1]*2 + 5;
8    }
9    printf("%d\n", tab[5]);
10   free(tab);
11   return 0;
12 }
```

a. tab = (int *) malloc(5*sizeof(int *));
b. tab = (int *) malloc(6*sizeof(int));
c. tab = (int) malloc(sizeof(6*int));
d. tab = (int) calloc(5, sizeof(int));

14. Que retourne la fonction *malloc* quand elle n'arrive pas à allouer un bloc de la taille demandée (par exemple quand la mémoire est pleine) ?

a. void
b. -1
c. NULL
d. on ne peut pas savoir

15. Que ce passe-t-il si l'on inverse les lignes 3 et 4 du code de la fonction f ci-dessous ?

```
1 void f(type* V) {
2   if (V != NULL) {
3     printf("%d\n",V->val);
4     f(V->next);
5   }
6 }
```

a. la fonction fera une boucle infinie,
b. l'affichage se fera en sens inverse,
c. la fonction n'affichera plus rien,
d. rien ne sera changé.

16. Laquelle des fonctions suivantes retourne le nombre de lettres de la chaîne de caractères passée en argument ?

a.
```
1 int strlen(char* m) {
2   int i;
3   while (m[i] == 0) {
4     i = i + 1;
5   }
6   return i;
7 }
```

b.
```
1 int strlen(char* m) {
2   int i = 1;
3   while (i > 0) {
4     i = m[i];
5   }
6   return i;
7 }
```

c.
```
1 int strlen(char* m) {
2   int i = 1;
3   while (i > 0) {
4     if (m[i] = 0) { i = 0;}
5   }
6   return m[i];
7 }
```

d.
```
1 int strlen(char* m) {
2   int i = 0;
3   while (m[i] != 0) {
4     i++;
5   }
6   return i;
7 }
```

17. Si ptr a été déclaré sous la forme int *ptr, un seul des groupes d'affirmations suivantes est vrai, lequel ?

a.	• l'expression *(ptr+1)==ptr[1] est syntaxiquement invalide • l'expression ptr[2] est une valeur entière • l'expression &ptr est une adresse en mémoire
b.	• l'expression *(ptr+1)==ptr[1] est toujours vraie • l'expression ptr-- est syntaxiquement invalide
c.	• l'expression *(ptr + 1) est une valeur entière • l'expression ptr[1] est une valeur entière • l'expression ptr est une adresse en mémoire

18. Considérons le code suivant :

```
int m[3][4]= { { 1, 2, 3, 4} ,
               { 5, 6, 7, 8} ,
               { 9, 10, 11, 12} };

printf("%p %d %d %d\n",m, m[1][2], m[0][5], m[2][2]);
```

Parmi les fragments de programme ci-dessous, un seul utilisant la notation avec les pointeurs est correct et affiche la même sortie. Lequel ?

a.
```
int *ptr=m;
printf("%p %d %d %d\n",ptr, *(ptr+4*1+2), *(ptr+4*0+5), *(ptr+2*4+2));
```

b.
```
int *ptr;

ptr=&(m[0][0]);
printf("%p %d %d %d\n",ptr, *(ptr+4*1+2), *(ptr+4*0+5), *(ptr+2*4+2));
```

c.	`int **ptr=m;` `printf("%p %d %d %d\n",ptr, *(ptr+4*1+2), *(ptr+4*0+5), *(ptr+2*4+2));`

19. Quelle est la différence entre un tableau et une structure?

a. Un tableau peut contenir des données de types différents, tandis qu'une structure ne le peut pas.
b. Une structure peut contenir des données de types différents, tandis qu'un tableau ne le peut pas.
c. Tous deux peuvent contenir des données de types différents, mais la structure occupe moins de place mémoire.
d. Tous deux peuvent contenir des données de types différents, mais la structure permet un accès mémoire plus rapide.

20. Considérons le fragment de code ci-dessous :

```
int tab[]={ 10, 20, 30, 40 };
int *ptr1=&tab[1];
int *ptr2=&tab[3];
```

Dans ce code, une seule des affirmations suivantes est vraie, laquelle ?

a. les expressions *(ptr1-1) et *(ptr2-3) retournent toutes les deux la même valeur, 10
b. l'expression ptr2-ptr1 vaut 20
c. l'expression *(ptr2-ptr1) retourne la valeur 20

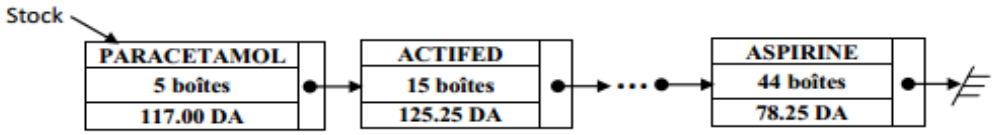
Partie III : Exercices de programmation en C *(10 points)*

Exercice 1 : Listes Chainées *(5,5 pts)*

Barème	***Question 1***	***Question 2***	***Question 3***	***Question 4***
	1 pt	***1,5 pts***	***1,5 pts***	***1,5 pts***

Un pharmacien souhaite traiter les informations concernant son stock de médicaments par ordinateur. On vous propose de représenter ces informations sous forme de liste linéaire chaînée où chaque maillon contient le libellé d'un médicament, la quantité disponible (nombre de boîtes) et le prix unitaire.

Exemple :



On vous demande de :

1. Donner les structures de données nécessaires à la représentation de ce stock.
2. Ecrire la procédure **Vendre(Med, NbBoites)** permettant de retirer, si possible, 'NbBoites' du médicament 'Med' du stock.
   Attention : Il faut supprimer du stock le médicament dont la quantité atteint 0.
3. Ecrire la procédure **Acheter(Med, NbBoites, Prix)** permettant au pharmacien d'alimenter son stock par 'NbBoites' du médicament 'Med' ayant le prix unitaire 'Prix' DA. On considère qu'un médicament prenne toujours le nouveau prix.
4. Ecrire la fonction **PrixStock** permettant de calculer le prix total des médicaments dans le stock.

Exercice 2 : Les structures *(4,5 pts)*

Un livre est caractérisé par :

- son numéro (entier)
- son titre (chaîne de caractère)
- son auteur (chaîne de caractère)
- son éditeur (chaîne de caractère)
- son prix (réel)

1. Ecrire le code qui définit un livre ***(1 pt)***
2. Ecrire une fonction qui permet de retourner le nombre d'exemplaires d'un livre donné ***(1 pt).***
3. Ecrire une fonction qui retourne le livre le plus cher.

4. Ecrire une fonction qui permet de calculer la valeur de la bibliothèque (somme des prix des livres) ***(1 pt)***
5. Ecrire une fonction qui permet de retourner une partie de la liste de livres à partir d'une position Deb à une position Fin. ***(1,5 pts)***

Programmation Structurée 2 : Examen Final

Grille de réponse au QCM

Question	Réponses						
	A	B	c	d	e	F	G
Q1				X			
Q2	X						
Q3		X					
Q4			X				
Q5			X				
Q6			X				
Q7			X				
Q8	X						
Q9						X	
Q10	X						
Q11	X						
Q12			X				
Q13		X					
Q14			X				
Q15		X					
Q16				X			
Q17			X				
Q18		X					
Q19		X					
Q20	X						