Université Internationale de Casablanca
Nous innovons pour votre réussite !

Ecole d'Ingénierie
Filières : CPI & MIAGE
Classe : 2ème année

Cours : Programmation
Professeur : MOUHAMID
Date : 02/12/2016

Partie I : QCM *(9 points)*

Répondez en entourant la/les lettre(s) correspondant(s) à la/les bonne(s) réponse(s).
+0,75 pour une bonne réponse, 0 pour absence de réponse, -0,25 pour une mauvaise réponse.

1. Donner le résultat de l'exécution du code suivant :

```
int i = 0;
int j = 1;
for (i = 0; i < 2; i = i + 2)
{
    for (j = 0; j < 3; j = j + 1)
    {
        printf("%d ", j);
    }
}
```

   a. 0 1 2 0 1 2
   b. 2 1 3 0 1 1
   c. 0 1 2
   d. 0 1 1
   e. Aucune des propositions ci-dessus.

2. L'allocation dynamique de mémoire suit des étapes dans un ordre particulier, lequel ?
   a. malloc, utilisation de la mémoire, free, vérification allocation réussie
   b. free, vérification de l'allocation réussie, malloc, utilisation de la mémoire
   c. malloc, utilisation de la mémoire, vérification allocation réussie, free
   d. malloc, vérification allocation réussie, utilisation de la mémoire , free

3. Combien de float peut-on stocker au maximum dans un tableau tab alloué avec la commande : `float* tab = (float*) malloc(4000);` ?

   a. 4000,
   b. 4000/sizeof(float)
   c. 4000*sizeof(float)
   d. On ne peut pas savoir

4. Laquelle des fonctions suivantes retourne le nombre de lettres de la chaîne de caractères passée en argument (Cochez la bonne réponse) ?

```
int strlen(char* m) {
  int i;
  while (m[i] == 0) {
    i = i + 1;
  }
  return i;
}
```

```
int strlen(char* m) {
  int i = 1;
  while (i > 0) {
    i = m[i];
  }
  return i;
}
```

```
int strlen(char* m) {
  int i = 1;
  while (i > 0) {
    if (m[i] = 0) { i = 0;}
  }
  return m[i];
}
```

```
int strlen(char* m) {
  int i = 0;
  while (m[i] != 0) {
    i++;
  }
  return i;
}
```

5. Donner le résultat de l'exécution du code suivant :

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

a. ne compile pas
b. provoque une erreur à l'exécution
c. affiche 0 à l'exécution
d. affiche 1 à l'exécution

6. Considérons le prototype de la fonction suivante: **void Fiche(float *X, float *Y,int i,char Z,char R)** ainsi que les déclarations suivantes :

float A, C ;
int J ;
char B, H ;

Quels sont les appels justes de la fonction Fiche?

a. Fiche (A,C ; J ; B, H) ;
b. Fiche (&A, &B, C, J, H) ;
c. Fiche (&A,&C, 3, 'b', B) ;
d. Fiche (&A, &C, J, B, H) ;
e. Fiche (A ; J ; B, H) ;

7. Donner le résultat de l'exécution 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 = 4, b = 5, *ptr1 = 5, *ptr2 = 5
d. affiche a = 4, b = 5, *ptr1 = 4, *ptr2 = 4
e. affiche a = 4, b = 4, *ptr1 = 5, *ptr2 = 4
f. Aucune des propositions ci-dessus.

8. Donner le résultat de l'exécution du code suivant :

```
int main() {
int a, b;
int *ptr1, *ptr2;
a = 6;
b = a+1;
ptr1 = &a;
a++;
ptr2 = &b;
```

```
b += a = b;
printf("a = %d, b = %d, *ptr1 = %d, *ptr2 = %d\n", a, b, *ptr1, *ptr2+1);
return 0;
}
```

a. ne compile pas
b. provoque une erreur à l'exécution (erreur de segmentation par exemple)
c. affiche a = 5, b = 10, *ptr1 = 5, *ptr2 = 10
d. affiche a = 7, b = 14, *ptr1 = 7, *ptr2 = 15
e. affiche a = 7, b = 14, *ptr1 = 7, *ptr2 = 7
f. Aucune des propositions ci-dessus.

9. Ce programme a un défaut. Mais lequel ?

```
main(){
    char ville[100];
    printf("Dans quelle ville habitez-vous ? ");
    scanf("%s", ville);
    printf("Vous habitez %s, je connais bien cette ville !", ville);
    }
```

a. Il manque un & devant la variable "ville" dans le printf
b. Il manque une * devant la variable "ville" dans la déclaration de la variable.
c. Il y a un & en trop devant "ville" dans le scanf.
d. Le programme ne contient aucune erreur.

10. Donner le résultat de l'exécution du code suivant :

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

```
    return 0;
}
```

a. ne compile pas
b. provoque une erreur fatale à l'exécution
c. affiche [4 2 2]
d. affiche [4 2 4]
e. affiche [5 3 5]
f. Aucune des propositions ci-dessus.

11. Donner le résultat de l'exécution du code suivant :

```
void fonction(int a[]) {
   a[2] = 5;
}
int main(void) {
   int T[]={11,22,33};
   fonction(T);
   printf("%d", T[2] );
   return 0;
}
```

a. 22
b. 11
c. 33
d. Le programme contient une erreur
e. 5

12. 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( ) {
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(6*sizeof(int));
b. tab = (int) malloc(sizeof(6*int));
c. tab = (int) calloc(5, sizeof(int));
d. tab = (int*) malloc(5*sizeof(int*));
e. Ce code contient des erreurs.

Partie II : Questions directes *(3 points)*

1. Un programme en C contient la déclaration suivante *(1 pt)*:

```
char *couleur[6]= {"rouge", "vert", "bleu", "blanc", "noir", "jaune"};
```

a. Quelle est la valeur de **couleur[3]** ?

.....bleu.....

b. Quelle est la valeur de ***(couleur[2] +2)** ?

.....Vert, blanc.....

2. Qu'affiche le programme suivant *(1 pt)* :

```
#include <stdio.h>

void affiche_chaine(char *T){
  int i=0;
  while(T[i]!='\0')
    {
      printf("%c",T[i]);
      i++;
    }
}
```

```
int main()
{
  char chaine[]={'e','x','a','m'};
  char tab[]="INGENIERIE";

  affiche_chaine(chaine);
  affiche_chaine(tab);

  return 0;
}
```

.....

.....

3. Ecrivez le code pour définir une **structure LIVRE** ayant les informations suivantes ***(1 pt)*** :
   - Le titre du livre (100 caractères),
   - Le nom de l'auteur (50 caractères),
   - L'année de son édition.

Partie III : Exercices de programmation *(8 points)*

Exercice 1 (4 points) - Tableaux & Allocations dynamiques de mémoire

Écrire programme qui :

- Créé deux tableaux t1 de taille N1 et t2 de taille N2 en optant pour l'allocation dynamique de mémoire (N1 et N2 sont définis par l'utilisateur).
- Rempli t1 et t2 avec des valeurs saisies par l'utilisateur.
- Agrandi la taille de t2 de telle façon à y rajouter les éléments de t1.
- Rajoute les éléments de le t2 à la fin de t1.
- Affiche t1.
- Libère les mémoires allouées.

Exercice 2 (4 points) – Tableaux de pointeurs& chaines de caractères

Ecrire un programme qui lit 10 mots d'une longueur maximale de 100 caractères au clavier et qui les mémorise dans un tableau de pointeurs sur char en réservant dynamiquement l'emplacement en mémoire pour les mots.
Ensuite, le programme vérifie s'il y a des mots palindromes et les affiche.