NOM & PRENOMS ……
……
SECTION …… GROUPE :……..

Université des Sciences et Technologie d'Oran USTO-MB
Faculté des MI, Département des MATHEMATIQUES
Année Universitaire : 2022-2023

EXAMEN

Algorithmique et structures de données 1
Durée 1h30

PARTIE COURS

Exercice 1 (7.5 points)

½ point QCM

1. Qu'affiche l'instruction suivante :
printf("j'aime le C %d fois", 30/20) ;
   - ☐ j'aime le C 0 fois
   - ☑ **j'aime le C 1 fois**
   - ☐ j'aime le C 1.5 fois
   - ☐ je n'aime pas le C
2. Qu'affichent les instructions suivantes :
int x=9 ; int y=x+10 ;
printf("%d :%d :%d",x,y,y) ;
   - ☐ 9 :19 :
   - ☑ **9 :19 :19**
   - ☐ 9 :19 :%d
   - ☐ %d :%d :%d
3. Qu'affichent les instructions suivantes :
char a ; a=65 ;
printf("%d %c",a,a) ;
   - ☐ A A
   - ☑ **65 A**
   - ☐ A 65
   - ☐ 65 65
4. Qu'affichent les instructions suivantes :
int i=10 ; while (i>0) {
i=i-4 ; printf("%d",i) ;
}
   - ☐ 1 0 6 2
   - ☑ **6 2 -2**
   - ☐ 6 2
   - ☐ C'est une boucle infinie
5. Si le nombre d'itérations est connu, il est conseillé d'utiliser :
   - ☐ while …
   - ☑ **for …**
   - ☐ do … while
6. L'instruction « switch » sert à éviter des instructions :
   - ☐ while … imbriquées
   - ☐ do … while imbriquées
   - ☑ **if … else … imbriquées**
   - ☐ for … imbriquées
7. On définit les constantes et les variables suivantes :
#define A 5
#define B 7
#define G 5.6
float c, d ;
int e=5, f=7
Lesquelles de ces déclarations sont-elles justes :
   - ☑ **int T1[10][B] ;**
   - ☑ **float T2[10][B] ;**
   - ☑ **int T3[A][B] ;**
   - ☐ int T4[G][B] ;
   - ☐ char T5[1][c] ;
   - ☐ float T6[e][f] ;
8. Lesquelles de ces déclarations de fonctions (signatures) sont correctes si elles sont censées calculer « x » élevé à la puissance « n » entière ?
   - ☑ **float puissance (float x, int n) ;**
   - ☐ puissance (float x, int n) ;
   - ☐ void puissance (float x, int n) ;
   - ☐ puissance() ;
9. Soient les lignes d'instruction suivantes :
struct timbre { int prix ;
int annee ;
char origine[20] ; char image[20] ;
} ;
struct timbre COLLECTION[10] ;

Comment accède-t-on à l'année du 3ème timbre de la collection ?
   - ☐ COLLECTION[2,2]
   - ☑ **COLLECTION[2].annee**
   - ☐ COLLECTION.annee[2]
   - ☐ COLLECTION.annee
   - ☐ (COLLECTION+2)->annee
10. Dans un programme C, **break** permet de :
   - ☑ **Sortir d'une boucle**
   - ☐ Arrêter un programme
   - ☑ **Sortir d'un block switch**
   - ☐ N'a aucun effet
11. Après l'instruction suivante : *char z=(2>1) ?'x' :'y' ;*
   - ☐ z est égale à 1
   - ☑ **z est égale à 'x'**
   - ☐ z est égale à 2,
   - ☐ z est égale à 'y'
12. On considère deux tableaux T1 et T2. Peut-on copier le contenu de T2 dans T1 sans perdre d'information ?
   - ☐ Directement si T1 et T2 sont de même taille ? On utilise l'instruction T1 =T2 ;
   - ☐ Directement si la taille de T1 est supérieure à la taille de T2 ? On utilise l'affectation T1 =T2 ;
   - ☐ Directement si la taille de T2 est supérieure à la taille de T1 ? On utilise l'affectation T1 =T2 ;
   - ☑ **Elément par élément à l'aide d'une boucle dès que la taille de T1 est >= à la taille de T2 ?**
13. En plus d'un (ou plusieurs) appel à lui-même, que doit toujours comporter un algorithme récursif ?
   - ☐ Une boucle for
   - ☐ Une étape de division
   - ☑ **Une condition de terminaison**
   - ☐ Une étape de fusion.
14. Quelle est la sortie du programme suivant ?
```
#include<stdio.h>
main()
{  int i = 1;
   while(++i <= 5)
   printf("%d ",i++);}
```
   - ☐ 1 3 5
   - ☑ **2 4**
   - ☐ 2 4 6
   - ☐ 2

15. Quelle valeur retourne la fonction **mystere** quand elle sera appelée avec une valeur égale à **4**?

```
fonction mystere(nb:entier) :entier
{ si(nb< = 1)alors
retourner(1);
sinon retourner(nb*mystere(nb–1)); }
```

- ☐ 0.
- ☐ 1.
- ☐ 4.
- ☑ **24**

Exercice 2 : (2.5 points)

1. Pour une procédure, son nom et la liste des paramètres formels se nomme :
   SIGNATURE
2. Qu'est ce qui indique qu'un sous-programme est une procédure et non une fonction ?
   Le type de retour vide ou Void dans C.
3. La solution récursive terminale est-elle plus rapide par rapport la récursivité classique ? Justifier.
   Oui car lors de l'appel récursive l'évaluation de résultat de calcul en paramètre est effectuée.
   Donc dans le retour aux environnements il n'y aura pas de calcul.
4. Expliquer les deux conditions nécessaires pour être en mesure d'utiliser la récursivité ?
   Il faut pouvoir exprimer un algorithme sous forme d'une fonction de telle manière que sa valeur à un certain rang ne dépende que de sa valeur aux rangs inférieurs.
   On doit aussi connaitre la solution pour les rangs initiaux.
5. Quel est le rôle des deux fonctions en langage C : *strcmp* et *strlen* ?
   strcmp : comparer deux chaînes.
   strlen : longueur d'une chaîne de caractère

½ point question

Exercice 3 : (4 points)

Un nombre **parfait** est un entier positif supérieur à 1, égal à la somme de ses diviseurs ; on ne compte pas comme diviseur le nombre lui-même.

Exemple : 6 est un nombre parfait puisque : 6 = 3 + 2 + 1.

1. Donner un nombre parfait différent de 6.
2. Ecrire la conception de l'algorithme qui nous dit si un entier n est parfait ou non.

1. Le nombre 28 est parfait puisque 28 = 14+4+7+2+1. ½ pt

2. **solution** :

1ère version:

```
Variables n, d, S : entier
Début
Ecrire ("Entrez la valeur de n : ")
Lire (n)
S ← 1
d ←2
TantQue (d*d <= n)
        Si (n%d=0) alors S ←S + d + n/d finsi
        d ←d + 1
FinTantQue
Si (S=n) alors
        Ecrire ("le nombre n est parfait ")
Sinon
        Ecrire ("le nombre n n'est pas parfait ")
finsi
Fin
```

½ pt

1.5 pt

1.5 pt

2ème version en c:

```
#include<stdio.h>
int main(){
int somme=0, nbr, i;
printf(" Entrez un nombre: ");
scanf("%d",&nbr);
for(i = 1; i < nbr; ++i){
if(nbr%i == 0){
somme = somme + i;
}
}
if(somme == nbr){
printf(" Nombre parfait");
}
else{
printf(" n'est pas un Nombre parfait");
}
return 0;
}
```

½ pt

1.5 pt

1.5 pt

Exercice 4 : (6 points)

1. Écrivez un algorithme qui lit la taille d'un tableau n, le tableau T, une valeur x, et il indique ensuite si l'élément x appartient ou non au tableau T.

```
variables n, i, x, T[20] : entiers
début
ecrire("entrer la valeur de n : ")
lire(n)
ecrire("entrer le tableau T : ")
pour i allant de 1 à n faire lire(T[i]) finpour
ecrire("entrer la valeur de x : ")
lire(x)
i ←1
TantQue ( i ≤ n et T[i] ≠ x) faire
        i ←i + 1
FinTantQue
Si (i ≠ n+1) alors
        Ecrire(" l'élément x se trouve dans le tableau T ")
Sinon
        Ecrire("l'élément x ne se trouve pas dans le tableau T ")
finsi
fin
```

0.5 pt

1.5 pt

1 pt

2. Écrivez un algorithme qui permet de déterminer si les éléments d'un tableau d'entiers sont tous consécutifs ou non. (Par exemple, si le tableau est : 7; 8; 9; 10, ses éléments sont tous consécutifs. Si le tableau est : 7; 9 ; 10; 11, ses éléments ne sont pas tous consécutifs).
Dans cette question, on suppose que les variables `n` et `T` sont connues.

1ère version:

```
Variable i: entier
Début
i ←1
TantQue (i<n ET T[i+1] = T[i]+1)
        i ←i+1
FinTantQue

Si (i<n) alors
        Ecrire("les termes du tableau ne sont pas consécutifs")
        Sinon
        Ecrire(" les termes du tableau sont consécutifs ")
        FinSi
        Fin
```

0.5 pt

1.5 pt

1 pt

2ème version:

```
Variable i, j: entier
Début
j ←1
Pour i allant de 1 à n-1
        Si (T[i+1] ≠ T[i] +1) alors
        j ←0
        i ←n-1
```

0.5 pt

1.5 pt

```
            FinSi
FinPour
Si (j=1) alors
            Ecrire("les termes du tableau sont consécutifs")
Sinon
            Ecrire("les termes du tableau ne sont pas consécutifs")
Finsi
Fin
```

1 pt