Nom et Prénom : _____

Corrigé Contrôle continu 2

Hicham Janati

1 Tutoriel (10 pts)

IMPORTANT : Chaque question doit être traitée sans utiliser des notions présentées dans les questions qui viennent après ! Créez un fichier .c avec lequel vous allez effectuer ce tutoriel. Dans tous les bouts de codes suivants, nous supposons **stdlib** et **stdio** inclus.

1. **char est un type d'entier conçu pour stocker des variables (à 8 bits : soit $2^8 = 256$ valeurs) entre -128 et 127.**

```c
int main(){
    char a;
    a = 67;
    printf("La variable a = %d\n",a);
    return 0;
}
```

Quelle est la sortie du programme ? Changez le "%d" par un "%c". Quelle est la sortie du programme ?

```
// La variable a = 67
// La variable a = C
```

2. **Essayez d'autres nombres voisins de 67. En déduire un moyen d'afficher la lettre K (en une ligne, mentionner le changement à effectuer au code).**

```
a = 75
```

3. **Bien évidemment on ne va pas apprendre le numéro de chaque caractère. Pour stocker un caractère en mémoire, il suffit d'ajouter des guillemets :**

```c
int main(){
    char a;
    a = 'A';
    printf("La variable a = %c\n",a);
    return 0;
}
```

Quelle est la sortie de ce programme ? Et si l'on change le '%c' en '%d' ? La variable a = A. La variable a = 65

4. **Ainsi, l'ordinateur interprète le type char comme étant un entier ou un caractère selon le type d'affichage donné à printf. Vous vous en doutiez sûrement : pour écrire en mémoire du texte, il suffit de créer un tableau de char : une chaîne de caractères ! Créez un tableau de char à 10 cases (non initialisé) qui contient un mot à 3 lettres de votre choix. (pas besoin de l'afficher)**

```c
int main(){
    char T[10];
    char[0] = 'Y';
    char[1] = 'E';
    char[2] = 'S';
    return 0;
}
```

5. **Comme avec les tableaux d'entiers où on initialisait avec des accolades comme *int T[4] = {0,1,3,4}*, on peut initialiser une chaîne de caractères avec des guillemets *char M[6]="hello"*. Complétez le code ci-dessous pour afficher les éléments de la chaîne un à un :**

```c
int main(){
    char M[16] = "Cool l'examen !";
    int i;
    for(i=0;i<16;i++)
        printf("%c",M[i]);
    return 0;
   }
```

6. **C'est un peu pénible non ? J'ai dû compter à la main le nombre de lettres pour créer mon tableau de taille 16 et itérer sur toute la chaîne pour l'afficher. Eh ben les programmeurs y ont déjà pensé : l'une des différences majeures entre tableaux et chaînes est qu'il n'est pas nécessaire de connaître la taille d'une chaîne ! Encore mieux, pour afficher une chaîne de caractère, il suffit de la donner à printf avec l'option %s. Essayez par vous-même :**

```c
int main(){
    char M[] = "Cool l'examen !";
    printf("%s", M);
    return 0;
    }
```

Mais comment C sait-il que le tableau est fini ? Eh ben en réalité, lorsqu'une chaîne est créée, C calcule le nombre de lettres, par exemple n mais réserve un tableau de taille n+1 en mémoire en ajoutant le caractère '\0' à la dernière case. Ce caractère signifie "STOP la chaîne se termine ici !". Ainsi, l'instruction *char H[]="hello";* crée un tableau à 6 cases : | h | e | l | l | o | \0 | La case supplémentaire H[5] vaut alors '\0'. Remarque : il faut toujours prévoir une case supplémentaire pour le '\0'. Remarquez à la question 5 que M contient 15 lettres mais j'ai réservé 16 cases en mémoire.

Comment peut-on itérer sur un tableau de char ? Pour calculer la taille de la chaîne M ci-dessus par exemple ? (Donner l'idée)

```c
// Une chaine de caractère est un tableau dont le dernier élément est toujours "\0"
// Ainsi, on peut parcourir ce tableau tant que le caractère actuel
// est différent de "\0"
```

7. **Complétez la fonction ci-dessous qui renvoie le nombre de caractère c trouvés dans une chaîne M :**

```c
int Compteur(char M[], char c){
    int i=0, nbr =0;
    while(M[i]!='\0'){
        // si le caractère actuel est c, on ajoute 1:
        if (M[i] == c) nbr ++;
        i++;
        }
```

```
8       return nbr;
9   }
```

8. **Compilez ce bout de code et expliquez l'erreur sortie dans le terminal :**

```
1   int main(){
2       char M[7] = "Serieux?";
3       return 0;
4       }
```

La taille du tableau est trop petite !

2 Problème (10 pts)

Le but de cet exercice est d'étudier les séries hyperharmoniques. On définit la série hyperharmonique d'ordre p par la somme :

$$S_p = \sum_{i=1}^{\infty} \frac{1}{i^p}$$

$p \leq 1 \Rightarrow S_p = +\infty$	$p > 1 \Rightarrow S_p < +\infty$	En particulier $S_2 = \frac{\pi^2}{6}$

Nous allons effectuer des approximations de cette somme avec une troncature à l'ordre N en sommant un tableau de taille N :

$$S_{p,N} = \sum_{i=1}^{N} \frac{1}{i^p}$$

Répondez aux questions suivantes en complétant les trois fichiers ci-dessous. (1pt pour la syntaxe)

1. Créez une fonction qui calcule la puissance p-ème d'un réel x. [bonus pour une forme récursive] (1.5pts)
2. Créez une fonction qui calcule la somme d'un tableau de réels. (2pts)
3. Créez une fonction qui prend un entier p, un entier N et vous renvoie une estimation de S_p. (3pts)
4. Créez un programme main qui demande à l'utilisateur de saisir p puis estime S_p avec N = 1000 et affiche sa valeur. (0.5pt)
5. Valeurs approchées de S_2 et S_3 ? (0.5pt)
6. En utilisant la fonction *log* de *math.h*, créez un programme qui donne une estimation de la constante gamma d'Euler (0.5pt) :

$$\gamma = \lim_{n \to \infty} (S_{1,N} - log(n))$$

7. En quoi l'utilisation des tableaux peut-elle être critiquée pour ce type de programme ? (1pt)

Fichier fonctions.h

```
1  float   Puissance(float x, int p);
2  float   SommeTableau(float T[], int N);
3  float   Estimation(int p, int N);
```

Fichier fonctions.c

```
1  float Puissance(float x, int p){
2      if (p==0)
3          return 0;
4      else
5          return x*Puissance(x,p-1);
6  }
```

```
7
8
9
10 float SommeTableau(float T[], int N){
11      int i;
12      float s=0;
13      for(i=0;i<N;i++){
14          s += T[i];}
15      return s;
16      }
17
18
19
20
21
```

```
22
23
24
25
26 float Estimation(int p, int N){
27     int i;
28     float T[N];
29     for(i=0;i<N;i++){
30         T[i] = 1/Puissance(i+1,p);
31         // i+1 car la somme commence à
   1 et finit à N
32         }
33     return SommeTableau(T,N);
34         }
35
36
37
38 .
```

Fichier main.c

```
1  #include<stdio.h>
2  #include<math.h>
3  #include"fonctions.h"
4  int main(){
5      int p,N=1000;
6      float s;
7      printf("Veuillez saisir p:\n");
8      scanf("%d",&p);
9      s = Estimation(p,N);
10     printf("s = %f\n", s);
11
12     // Pour Q6:
13     float gamma;
14     gamma =  Estimation(1,N) - log(N);
15     printf("gamma = %f\n", gamma );
16     return 0;
17    }
```

Q5-Q6-Q7

```
1  //  S2 = 1.645
2  //  S3 = 1.202
3  //  Gamma = 0.578
```

Ici, on ne s'intéresse qu'à la somme du tableau. Il est donc inutile de stocker toutes les cases en mémoire ! On peut tout simplement sommer avec une boucle directement :

```
1 float Estimation_2(int p, int N){
2         int i;
3     float s =0;
4     for(i=0;i<N;i++)
5         s+= 1/Puissance(i+1,p);
6     return s;
7     }
```

3 Exercices Bonus

Écrire une fonction qui renvoie le min d'un tableau de manière récursive. (3pts)
Écrire une fonction qui renvoie le nombre de doublures d'un tableau *Exemple :* |4|5|3|3| *: 1* |3|1|0|0|0|1| *: 2* (3pts)

```
1  float min_recursive(float T[],int N){
2      /* si le tableau ne contient qu'une case, on renvoie sa valeur */
3      if (N==1) return T[N];
4      /* sinon, on compare la dernière case avec le tableau de taille N-1*/
5      float min_davant = min_recursive(T,N-1);
6      if (min_davant < T[N-1])
7          return min_davant;
8      else
9          return T[N-1];
10     }
```

```c
int doublures(int T[], int N){
    int i=0, doublures=0;
    /* Il faut d'abord trier le tableau avec
    une fonction de tri à coder */

    Tri(T,N);

    /* Maintenant, on parcourt le tableau en "sautant" les cases identiques. */

    while(i<N){
        if (T[i] == T[i+1]) doublures ++;
        while(T[i] == T[i+1]) i++;
        //cette boucle nous permet de sauter tout le bloc identique
        i++;
    }
    return doublures;
}
```