Matière : Algorithmique et structures de données I
Date : 10 Janvier 2024
Durée : 1h30
Semestre 1

Université Abou Bakr Belkaïd - Tlemcen
Faculté des Sciences
Département de Mathématiques
L1-Maths-MI

Examen final

Les appareils portables doivent être éteints
Les solutions doivent être rédigées en **C**

1 Affichage (10 pts)

Qu'affichent les deux programmes suivants :

1.

```
#include <stdio.h>
#include <stdlib.h>
void main(){
 int i, rows=5, k=1;
 for(i=1 ; i<=rows*2 ; i+=2)
   { if (k%2)  printf("%3d %3d", i, i+1);
        else   printf("%3d %3d", i+1, i);
       k++;
     printf("\n");
} }
```

2.

```
#include <stdio.h>
#include <stdlib.h>
int fct1(int);  int fct2(int);  int fct3(int);
void main() {
 int val=12345, X, Y, Z;
 X=fct1(val);
 printf("……\n", val, X);
 Y=fct2(val);
 printf("……\n", val, Y);
 Z=fct3(val);
 printf("……\n", val, Z);
}
```

```
int fct1(int N)
{ int nbr=0;
  while(N) {
        nbr++;
        N/=10;  }
 return nbr;
}
int fct2(int N)
{ int a=0;
  while(N>0) {
        a+=N%10;
        N=N/10;  }
 return a;
}
int fct3(int N)
{ int V,nbr;
 nbr=N;
 do{  V=(V*10)+nbr%10;
      nbr/=10;
    } while(nbr>0);
 return V;
}
```

* Deviner ce que font les fonctions **fct1**, **fct2** et **fct3.**
* Compléter les différents commentaires devant chaque instruction **printf**.

2 Suite aliquote (10 pts)

- La suite aliquote est une séquence de nombres où chaque terme est la somme des diviseurs propres du nombre précédent (sauf le nombre lui-même).

Exemple **:** pour générer la suite aliquote à partir du nombre 12, on suit les étapes suivantes :

Étape 0: Le nombre initial est 12.
Étape 1: La somme des diviseurs propres de 12 est 1 + 2 + 3 + 4 + 6 = 16.
Étape 2: On calcule la somme des diviseurs propres de 16. La somme est 1 + 2 + 4 + 8 = 15.
Étape 3: La somme des diviseurs propres de 15 est 1 + 3 + 5 = 9.
Étape 4: La somme des diviseurs propres de 9 est 1 + 3 = 4.
Étape 5: La somme des diviseurs propres de 4 est 1 + 2 = 3.
Étape 6: La somme des diviseurs propres de 3 est 1.
À ce stade, on atteint 1, et la suite aliquote se termine.

Ainsi, la suite aliquote générée à partir du nombre 12 est : 12, 16, 15, 9, 4, 3, 1.

- L'ordre d'une suite aliquote est déterminé par le nombre d'itérations nécessaires pour que la suite atteigne 1.
- Une suite générer de cette façon peut être considérée comme non-aliquote si elle entre dans un cycle récurrent sans fin, cela signifie qu'elle ne pourra pas atteindre 1 et donc elle est considérée comme non-aliquote.

Exemples :

* La suite aliquote formée par le nombre 12 atteint 1 en 6 étapes. Donc, l'ordre de cette suite aliquote est de 6.
* La suite aliquote générée à partir du nombre 95 est : 95, 25, 6, 6, (suite répétitive).
  Dans ce cas, la suite se répète à partir du terme 6, et elle n'atteint jamais 1. Par conséquent, elle est considérée comme non-aliquote et son ordre ne peut pas être déterminé car elle entre dans un cycle.
* La suite aliquote générée à partir du nombre 1184 est : 1184, 1210, 1184 (Suite répétitive).
  Ici, la suite aliquote commence à boucler sur le nombre lui-même, revenant à 1184 après avoir atteint 1210. Cela crée un cycle où la suite répète le nombre initial, ne progressant pas davantage vers d'autres valeurs.

Écrire un programme qui génère et affiche la suite aliquote pour un nombre entier donné, ainsi que son ordre. Pour cela, le programme doit contenir les fonctions suivantes :

1. La fonction **Lire_n** qui lit la valeur d'un nombre, si ce nombre est négatif, la fonction redemande un entier. Elle retournera la valeur entrée une fois qu'elle sera valide.
2. La fonction **SommeDiviseurs** qui calcule la somme des diviseurs propres (les diviseurs autres que le nombre lui-même) d'un nombre donné et renvoie cette somme.
3. La fonction **GenererSuiteAliquote** qui génère la suite aliquote du nombre qui lui est passé en paramètre. Elle affiche chaque élément de la suite et détermine son ordre.
   * Si la suite atteint 1, elle affiche la suite et retourne l'ordre.
   * Si la suite entre dans un cycle ou dépasse un certain nombre d'itérations (par exemple : 1000), elle retourne 0.
4. **VerifierAliquoteness :** Cette fonction appelle **GenererSuiteAliquote** pour vérifier si la suite est une suite aliquote. Si c'est le cas, elle affiche que la suite est une suite aliquote avec son ordre. Sinon, elle affiche que la suite est non-aliquote.
5. Le programme principal doit demander à l'utilisateur d'entrer un nombre entier positif, puis utiliser la fonction **VerifierAliquoteness** pour déterminer si la suite générée à partir de ce nombre est une suite aliquote.

« Bon courage »

Final Exam

Portable devices should be turned off
Solutions should be written in **C**

1 Display (10 pts)

What do the following two programs display:

1.
```
#include <stdio.h>
#include <stdlib.h>
void main(){
 int i, rows=5, k=1;
 for(i=1 ; i<=rows*2 ; i+=2)
   { if (k%2)  printf("%3d %3d", i, i+1);
       else   printf("%3d %3d", i+1, i);
      k++;
    printf("\n");
} }
```

```
int fct1(int N)
{ int nbr=0;
  while(N) {
        nbr++;
        N/=10;  }
 return nbr;
}
int fct2(int N)
{ int a=0;
  while(N>0) {
        a+=N%10;
        N=N/10;  }
 return a;
}
int fct3(int N)
{ int V,nbr;
  nbr=N;
  do{  V=(V*10)+nbr%10;
       nbr/=10;
     } while(nbr>0);
  return V;
}
```

2.
```
#include <stdio.h>
#include <stdlib.h>
int fct1(int);  int fct2(int);  int fct3(int);
void main() {
 int val=12345, X, Y, Z;
 X=fct1(val);
 printf("……\n", val, X);
 Y=fct2(val);
 printf("……\n", val, Y);
 Z=fct3(val);
 printf("……\n", val, Z);
}
```

* Guess what the functions **fct1**, **fct2** and **fct3** do.
* Complete the various comments of each instruction **printf**.

2 Aliquot sequence (10 pts)

- The aliquot sequence is a sequence of numbers where each term is the sum of the eigen divisors of the preceding number (except the number itself).

Example **:** To generate the aliquot sequence from the number 12, follow these steps:

Step 0: The initial number is 12.
Step 1: The sum of the eigen divisors of 12 is 1 + 2 + 3 + 4 + 6 = 16.
Step 2: We calculate the sum of the eigen divisors of 16. The sum is 1 + 2 + 4 + 8 = 15.
Step 3: The sum of the eigen divisors of 15 is 1 + 3 + 5 = 9.
Step 4: The sum of the eigen divisors of 9 is 1 + 3 = 4.
Step 5: The sum of the eigen divisors of 4 is 1 + 2 = 3.
Step 6: The sum of the eigen divisors of 3 is 1.
At this point, we reach 1, and the aliquot sequence ends.

Thus, the aliquot sequence generated from the number 12 is: 12, 16, 15, 9, 4, 3, 1.

- The order of an aliquot sequence is determined by the number of iterations required for the sequence to reach 1.
- The sequence generated in this way may be considered non-aliquot if it enters an endless recurring cycle, meaning that it cannot reach 1 and is, therefore, considered non-aliquot.

Examples :

* The aliquot sequence formed by the number 12 reaches 1 in 6 steps. So the order of this aliquot sequence is 6.
* The aliquot sequence generated from the number 95 is : 95, 25, 6, 6, (Repetitive sequence).
  In this case, the sequence is repeated from term 6, and it never reaches 1. Therefore, it is considered non-aliquot, and its order cannot be determined because it enters a cycle.
* The aliquot sequence generated from the number 1184 is: 1184,1210,1184, (Repetitive sequence).
  Here, the aliquot sequence begins to loop on the number itself, returning to 1184 after reaching 1210. This creates a cycle where the sequence repeats the initial number, making no further to progress towards other values.

Write a program that generates and displays the aliquot sequence for a given integer, as well as its order. To do this, the program must contain the following functions:

1. The function **read_n** which reads the value of a number, if that number is negative, the function requests an integer again. It will return the entered value once it is valid.
2. The function **SumDivisors**, that calculates the sum of the eigen divisors (the divisors other than the number itself) of a given number and returns this sum.
3. The function **GenerateSuiteAliquot**, which generates the aliquot sequence of the number passed to it as a parameter. It displays each element of the sequence and determines its order.
   * If the sequence reaches 1, it displays the sequence and returns the order.
   * If the sequence enters a cycle or exceeds a certain number of iterations (e.g. 1000), it returns 0.
4. **CheckAliquotness**: This function calls **GenerateSuiteAliquot** to check if the sequence is an aliquot sequence. If so, it displays that the sequence is an aliquot sequence with its order. Otherwise, it displays that the sequence is non-aliquot.
5. The main program should ask the user to enter a positive integer, and then use the **CheckAliquotness** function to determine whether the sequence generated from this number is an aliquot sequence.

« Good luck »

Matière : Algorithmique et structures de données I
Date : 10 Janvier 2024
Durée : 1h30
Semestre 1

Université Abou Bakr Belkaïd - Tlemcen
Faculté des Sciences
Département de Mathématiques
L1-Maths-MI

Corrigé Examen Final

1 Affichage (10 pts)

Qu'affichent les deux programmes suivants :

1.

```
#include <stdio.h>
#include <stdlib.h>
void main(){
  int i, rows=5, k=1;
  for(i=1 ; i<=rows*2 ; i+=2)
    { if (k%2)  printf("%3d %3d", i, i+1);
        else   printf("%3d %3d", i+1, i);
       k++;
     printf("\n");
} }
```

Solution Affichage1(**4pts**)

```
 1  2
 4  3
 5  6
 8  7
 9 10
```

(4pts)
0.25 /valeur
1.5pt pour le format :
(saut de ligne et espaces)

2.

```
#include <stdio.h>
#include <stdlib.h>
int fct1(int);  int fct2(int);  int fct3(int);
void main() {
  int val=12345, X, Y, Z;
  X=fct1(val);
  printf("……\n", val, X);
  Y=fct2(val);
  printf("……\n", val, Y);
  Z=fct3(val);
  printf("……\n", val, Z);
}
```

```
int fct1(int N)
{ int nbr=0;
   while(N) {
          nbr++;
          N/=10;  }
 return nbr;
}
int fct2(int N)
{ int a=0;
   while(N>0) {
          a+=N%10;
          N=N/10;  }
 return a;
}
int fct3(int N)
{ int V,nbr;
  nbr=N;
  do{  V=(V*10)+nbr%10;
       nbr/=10;
     } while(nbr>0);
  return V;
}
```

Solution Affichage2(**6pts**)

```
Le nombre de chiffre de 12345 est: 5
La somme de tous les chiffres de 12345 est: 15
L'inverse du nombre 12345 est: 54321
```

1 pour chaque résultat **(3pts)**

* Deviner ce que font les fonctions **fct1**, **fct2** et **fct3.**
  fct1 : Retourne le nombre de chiffre du nombre passé en paramètre. **(0.5pt)**
  fct2 : Retourne la somme de tous les chiffres du nombre passé en paramètre. **(0.5pt)**
  fct3 : Retourne l'inverse du nombre passé en paramètre. **(0.5pt)**
* Compléter les différents commentaires devant chaque instruction **printf**. **(1.5pts)**

```
1. printf("Le nombre de chiffre de %d est: %d\n", val, X);
2. printf("La somme de tous les chiffres de %d est: %d\n", val, Y);
3. printf("L'inverse du nombre %d est: %d\n", val, Z);
```

0.5 pour chaque commentaire (la présence des deux %d)

2 Suite aliquote (10 pts)

- La suite aliquote est une séquence de nombres où chaque terme est la somme des diviseurs propres du nombre précédent (sauf le nombre lui-même).

Exemple **:** pour générer la suite aliquote à partir du nombre 12, on suit les étapes suivantes :

Étape 0: Le nombre initial est 12.
Étape 1: La somme des diviseurs propres de 12 est 1 + 2 + 3 + 4 + 6 = 16.
Étape 2: On calcule la somme des diviseurs propres de 16. La somme est 1 + 2 + 4 + 8 = 15.
Étape 3: La somme des diviseurs propres de 15 est 1 + 3 + 5 = 9.
Étape 4: La somme des diviseurs propres de 9 est 1 + 3 = 4.
Étape 5: La somme des diviseurs propres de 4 est 1 + 2 = 3.
Étape 6: La somme des diviseurs propres de 3 est 1.
À ce stade, on atteint 1, et la suite aliquote se termine.
Ainsi, la suite aliquote générée à partir du nombre 12 est : 12, 16, 15, 9, 4, 3, 1.

- L'ordre d'une suite aliquote est déterminé par le nombre d'itérations nécessaires pour que la suite atteigne 1.
- Une suite générer de cette façon peut être considérée comme non-aliquote si elle entre dans un cycle récurrent sans fin, cela signifie qu'elle ne pourra pas atteindre 1 et donc elle est considérée comme non-aliquote.

Exemples :

* La suite aliquote formée par le nombre 12 atteint 1 en 6 étapes. Donc, l'ordre de cette suite aliquote est de 6.
* La suite aliquote générée à partir du nombre 95 est : 95, 25, 6, 6, (suite répétitive).
Dans ce cas, la suite se répète à partir du terme 6, et elle n'atteint jamais 1. Par conséquent, elle est considérée comme non-aliquote et son ordre ne peut pas être déterminé car elle entre dans un cycle.
* La suite aliquote générée à partir du nombre 1184 est : 1184, 1210, 1184 (Suite répétitive).
Ici, la suite aliquote commence à boucler sur le nombre lui-même, revenant à 1184 après avoir atteint 1210. Cela crée un cycle où la suite répète le nombre initial, ne progressant pas davantage vers d'autres valeurs.

Écrire un programme qui génère et affiche la suite aliquote pour un nombre entier donné, ainsi que son ordre. Pour cela, le programme doit contenir les fonctions suivantes :

1. La fonction **Lire_n** qui lit la valeur d'un nombre, si ce nombre est négatif, la fonction redemande un entier. Elle retournera la valeur entrée une fois qu'elle sera valide.

(1.5pts)

```
int Lire_n(){
 int n;
 do{
   printf("Entrez un nombre entier positif : ");
   scanf("%d", &n);
   } while(n<=0);
return n;
}
```

2. La fonction **SommeDiviseurs** qui calcule la somme des diviseurs propres (les diviseurs autres que le nombre lui-même) d'un nombre donné et renvoie cette somme.

(2pts)

```
int SommeDiviseurs(int n) {
int somme=0;
   for (int i=1; i<=n/2; ++i)
      if (n%i == 0)  somme += i;
return somme;
}
```

3. La fonction **GenererSuiteAliquote** qui génère la suite aliquote du nombre qui lui est passé en paramètre. Elle affiche chaque élément de la suite et détermine son ordre.
   * Si la suite atteint 1, elle affiche la suite et retourne l'ordre.
   * Si la suite entre dans un cycle ou dépasse un certain nombre d'itérations (par exemple : 1000), elle retourne 0.

(3.5pts)

```
int GenererSuiteAliquote(int n) {
  int valeur=n, precedent, ordre=0;

  printf("Suite aliquote pour %d : %d  ",n,valeur);
  do{  precedent = valeur;
      valeur = sommeDiviseurs(valeur);
      printf("%d  ",valeur);
      ordre++;
      if (valeur==n || valeur==precedent || ordre>1000 )          // Limite pour éviter les boucles infinies
           { printf("Suite repetitive\n");
             return 0;}                          // Si la suite se répète sans atteindre 1, retourne 0
    } while (valeur!=1);

return ordre;      // Si la suite atteint 1, retourne l'ordre
}
```

Remarque : D'autres versions peuvent être proposées.

4. **VerifierAliquoteness :** Cette fonction appelle **GenererSuiteAliquote** pour vérifier si la suite est une suite aliquote. Si c'est le cas, elle affiche que la suite est une suite aliquote avec son ordre. Sinon, elle affiche que la suite est non-aliquote.

(2pts)

```
void VerifierAliquoteness(int n) {
   int ordre = genererSuiteAliquote(n);

   if (ordre != 0) {
      printf("\nLa suite est une suite aliquote");
      printf("\nL'ordre de la suite est : %d\n", ordre);}
    else
      printf("\nLa suite n'est pas une suite aliquote\n");
}
```

5. Le programme principal doit demander à l'utilisateur d'entrer un nombre entier positif, puis utiliser la fonction **VerifierAliquoteness** pour déterminer si la suite générée à partir de ce nombre est une suite aliquote.

(1pt)

```
int main() {
   int nombre;
   nombre=read_n();
   verifierAliquoteness(nombre);
 // ou bien:    verifierAliquoteness(read_n());
return 0;
}
```

Final Exam

Portable devices should be turned off
Solutions should be written in **C**

1 Display (10 pts)

What do the following two programs display:

1.

```
#include <stdio.h>
#include <stdlib.h>
void main(){
 int i, rows=5, k=1;
 for(i=1 ; i<=rows*2 ; i+=2)
   { if (k%2)  printf("%3d %3d", i, i+1);
        else   printf("%3d %3d", i+1, i);
       k++;
     printf("\n");
} }
```

```
int fct1(int N)
{ int nbr=0;
  while(N) {
        nbr++;
        N/=10;  }
 return nbr;
}
int fct2(int N)
{ int a=0;
  while(N>0) {
        a+=N%10;
        N=N/10;  }
 return a;
}
int fct3(int N)
{ int V,nbr;
  nbr=N;
  do{  V=(V*10)+nbr%10;
       nbr/=10;
     } while(nbr>0);
  return V;
}
```

2.

```
#include <stdio.h>
#include <stdlib.h>
int fct1(int);  int fct2(int);  int fct3(int);
void main() {
 int val=12345, X, Y, Z;
 X=fct1(val);
 printf(".....\n", val, X);
 Y=fct2(val);
 printf(".....\n", val, Y);
 Z=fct3(val);
 printf(".....\n", val, Z);
}
```

* Guess what the functions **fct1**, **fct2** and **fct3** do.
* Complete the various comments of each instruction **printf**.

2 Aliquot sequence (10 pts)

- The aliquot sequence is a sequence of numbers where each term is the sum of the eigen divisors of the preceding number (except the number itself).

Example **:** To generate the aliquot sequence from the number 12, follow these steps:

Step 0: The initial number is 12.
Step 1: The sum of the eigen divisors of 12 is 1 + 2 + 3 + 4 + 6 = 16.
Step 2: We calculate the sum of the eigen divisors of 16. The sum is 1 + 2 + 4 + 8 = 15.
Step 3: The sum of the eigen divisors of 15 is 1 + 3 + 5 = 9.
Step 4: The sum of the eigen divisors of 9 is 1 + 3 = 4.
Step 5: The sum of the eigen divisors of 4 is 1 + 2 = 3.
Step 6: The sum of the eigen divisors of 3 is 1.
At this point, we reach 1, and the aliquot sequence ends.

Thus, the aliquot sequence generated from the number 12 is: 12, 16, 15, 9, 4, 3, 1.

- The order of an aliquot sequence is determined by the number of iterations required for the sequence to reach 1.
- The sequence generated in this way may be considered non-aliquot if it enters an endless recurring cycle, meaning that it cannot reach 1 and is, therefore, considered non-aliquot.

Examples :

* The aliquot sequence formed by the number 12 reaches 1 in 6 steps. So the order of this aliquot sequence is 6.
* The aliquot sequence generated from the number 95 is : 95, 25, 6, 6, (Repetitive sequence).
  In this case, the sequence is repeated from term 6, and it never reaches 1. Therefore, it is considered non-aliquot, and its order cannot be determined because it enters a cycle.
* The aliquot sequence generated from the number 1184 is: 1184,1210,1184, (Repetitive sequence).
  Here, the aliquot sequence begins to loop on the number itself, returning to 1184 after reaching 1210. This creates a cycle where the sequence repeats the initial number, making no further to progress towards other values.

Write a program that generates and displays the aliquot sequence for a given integer, as well as its order. To do this, the program must contain the following functions:

1. The function **read_n** which reads the value of a number, if that number is negative, the function requests an integer again. It will return the entered value once it is valid.
2. The function **SumDivisors**, that calculates the sum of the eigen divisors (the divisors other than the number itself) of a given number and returns this sum.
3. The function **GenerateSuiteAliquot**, which generates the aliquot sequence of the number passed to it as a parameter. It displays each element of the sequence and determines its order.
   * If the sequence reaches 1, it displays the sequence and returns the order.
   * If the sequence enters a cycle or exceeds a certain number of iterations (e.g. 1000), it returns 0.
4. **CheckAliquotness**: This function calls **GenerateSuiteAliquot** to check if the sequence is an aliquot sequence. If so, it displays that the sequence is an aliquot sequence with its order. Otherwise, it displays that the sequence is non-aliquot.
5. The main program should ask the user to enter a positive integer, and then use the **CheckAliquotness** function to determine whether the sequence generated from this number is an aliquot sequence.

« Good luck »