Université Libanaise
Faculté des Sciences III
Tripoli – Liban

Cours: **I1101**
Examen : **Final**

Date : **31/7/ 2019**
Durée : **2 Heures**

Exercice I : (25 points)

1- Ecrire la fonction **puissance** (float x, int n) qui calcule et retourne $\mathbf{x^n}$;
2 - Ecrire la fonction **factoriel** (n) qui calcule et retourne **n!** ;
Ecrire un programme principal qui utilise les deux fonctions précédentes pour calculer et retourner la valeur du e^x en additionnant seulement les 20 premiers termes de sa formule du développement limités de e^x ;
Remarque : $(e^x = 1 + x/1! + x^2/2! + x^3/3! + \ldots\ldots + x^{20}/20)$.

Exercice II (20 points)

Un nombre N (de trois chiffres) est dit Armstrong s'il est égal la somme de cube de ses chiffres.
*par exemple : 153 = 1*1*1 + 5*5*5 + 3*3*3 ;*

1 – Ecrire un fonction **Armstrong** (int k) qui teste si k est un nombre Armestrong ou non ;
2- Ecrire un programme principal qui nous affiche tous les nombres Armestong (de trois chiffres) en utilisant la fonction précédente.

Exercice III : (20 points)

Un nombre N est dit parfait s'il est égal la somme de ses diviseurs différents de N. Ecrire une fonction ***parfait*** *(...)* qui prend un entier n comme paramètre d'entrée et qui retourne **true** (ou **1**) si n est un nombre parfait , et sinon retourne **false (ou 0)** .
Ecrire un programme qui affiche tous les nombres parfaits entre 1000 et 10000.
(Note : le programme doit appeler la fonction ***parfait***(..) plusieurs fois.)
Exemples des nombres parfaits:
(i) diviseurs de 6 sont 1 , 2 , 3 et $6 = 1 + 2 + 3$
(ii) diviseurs de 28 sont 1 ,2 , 4 , 7, 14 et $28 = 1 + 2 + 4 + 7 + 14$

Exercice IV (35 points)

On considère un tableau T [ ] qui contient 20 entiers. Ecrire les fonctions suivantes :

1. *lire_Tab(...)* qui permet de lire le tableau T [ ].
2. *Repetition (int T[], int x...)* qui prend un tableau T [ ] et l'entier x comme paramètres et renvoie le nombre d'occurrence de x dans le tableau T[] ;
3. *teste_croissant(int T[])* qui retourne 1 si les éléments de T[] sont en ordre croissant et retourne 0 sinon ;
4. *teste_ensemble (int T[])* qui prend comme argument le tableau T[] et teste si le tableau T est un ensemble ou non (T est un ensemble si tous les éléments de T sont distincts).

Ecrire un programme en C qui fait appel aux fonctions précédentes et affiche leurs résultats.

Bon travail

Université Libanaise
Faculté des Sciences III
Tripoli – Liban

Cours: **I1101**
Examen : **Final +Partiel**

Date : **30 Juin 2018**
Durée : **2h et 30m**

Partie P (partiel)

Exercice I : (50 points)

Soit le code en C de la fonction suivante :

```
int  travail  (int  n)
    { int  s, i=0 ;
      s=0 ;
      While  (n != 0)
        {    s = s +n % 10 ;
             n = n / 10 ;
             i ++ ;
        }
      printf (" nb = %d \n", i ) ;
      return  s ;
    }
```

- Exécuter la fonction travail (785402),
- Que fait cette fonction en général ?

Exercice II : (50 points)

Ecrire un programme qui lit un entier n tel que $4 < n < 15$ et qui dessine un trapèze comme suit :

```
*  *  *  *  *  *
*  *  *  *  *  *  *
*  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *  *  *  *
```

L'exemple est fait pour $n = 6$.

Partie F (final)

Exercice III : (20 points)

Ecrire un programme qui lit un entier positif n et qui calcule et affiche la somme suivante :

$$S_n = \sum_{k=0}^{n} \left(\frac{5k+1}{2(k*k)+3} \right)$$

Exercice IV : (20 points)

Quel résultat affiche ce programme

```
void main ()
  { int sum =0 ;
    int x =1 ;
    int i ;
    for (i=0; i<= 4; i++)
    {    sum = sum +x;
         x = 2*x;
    }
    printf (" %d \n", sum );
}
```

Exercice V : (60 points)

On considère un tableau A [ ] qui contient 10 entiers. Ecrire les fonctions suivantes :

1. *lire_Tab(...)* qui permet de saisir le tableau A [ ].
2. *Moyenne_Tab(...)* qui prend le tableau A [ ] comme paramètre et renvoie la moyenne des éléments de A [ ].
3. *teste_croissant(...)* qui prend le tableau A [ ] comme paramètre et retourne 1 si les éléments de A[] sont en ordre croissant et retourne 0 sinon ;
4. *renverse_Tab (..)* qui prend comme le tableau A[] comme paramètre et renverse ses éléments c.-à-d.le premier élément devient le dernier et le dernier devient le premier et ainsi de suite (sans utiliser un deuxième tableau); *Ex : si A = {0,1.2,3,4,5}* le tableau demande devient *A ={5,4,3,2,1,0}* ;
5. *remplace(..:)* qui prend comme le tableau A[] comme paramètre et remplace chaque élément négatif par le nombre des éléments qui le précèdent ;
   Ex : si A= {3, 1, -4, 2, -8} le tableau devient A= {3, 1, 2, 2, 4} ;
6. *repetition_max(...)* qui prend le tableau A comme paramètre et retourne le nombre de répétition de l'élément maximum de A ;

crire un programme en C qui fait appel aux fonctions précédentes et affiche leurs résultats.

Bon travail

2017 - 2018 1ère session I1101

Partie Final :

Ex III :

```
#include <stdio.h>
void main(){
int i ;
float s=0;
do { printf("donner n+");
     scanf("%d",n);
   }
while (n<=0)
for(i=0; i<n, i++)
s= s+ ((5*i)+1)*1.0)/((2*i*i)+3);
printf("la somme est %f", s);
}
```

Ex V :

(1)* void lire tab (int V[])
{int i ;
for(i=0; i<10; i++)
{printf("donner V[%d]", i);
scanf("%d", &V[i]) ;}
}

(2)* float moyenne (int U[])
{int i, s=0;
for(i=0; i<10; i++)
s= s+ U[i];
return (float) s/10;}

(3)*
```
int teste-croissant(int t[])
{int i, ok=1;
for(i=0; i<9 ;i++)
 if(t[i]>t[i+1]) ok=0;
  return ok;}
```

2ème Méthode

```
{int i = 0;
while(t[i]<=t[i+1] && i<9)
      i++;
if (t[i]>t[i+1]) return 0;
                 else return 1;
}
```

(4)*
```
void reverse(int A[])
{int i, M;
 for(i=0; i<5; i++)
 {M = A[i];
  A[i] = A[9-i];
   A[9-i] = M;}
}
```

(5)*
```
Void remplace(int T[]){
int i;
for(i=0; i<10; i++)
if(T[i]<0) T[i] = i;
}
```

(6)*
```
int répétition(int T[])
{int i, M=t[0], K=1;
for(i=0, i<10, i++)
if(T[i]>M){M=t[i]; K=1;}
   else if(t[i]==M) K++;
  return K;}
```

progra e principal

```
#include <stdio.h>
  void main()
{int T[10];
lire_tab(t);
if (Teste_croissant(t) == 1)
  printf("les élts sont en ordre
          croissant \n");
else
  printf("les élts ne sont pas en ordre
          croissant \n"
renverse(t)
printf("la moyenne est: %f \n", )
          moyenne(t)
remplace(t);
printf("répétition de max est: %d \n",
          répétition_max(t)
}
```

Université Libanaise
Faculté des Sciences III
Tripoli – Liban

Cours: **I1101**
Examen : **Final**

Date : 18/9/2018
Durée : **2h**

Exercice I : (20 points)

Soit le code en C de la fonction suivante :

```
int  fct (int  n)
     { int  s, i=0 ;
       s=0 ;
       While  (n != 0)
        { if (n % 10  == 0 )  { s = s +n % 10 ;
                                i ++ ;}
          n = n / 10 ;}

       printf (" nb = %d \n", i ) ;
       return  s ;
     }
```

- Exécuter la fonction **fct** (9635210),
- Que fait cette fonction en général ?

Exercice II : (20 points)

Ecrire un programme qui lit un entier n tel que $3 < n < 15$ et qui dessine un triangle comme suit :

L'exemple est fait pour n = 4.

Exercice III : (20 points)

Ecrire un algorithme qui demande à l'utilisateur de taper un entier n et qui calcule U(n) défini par :

$U(0) = 3$
$U(n+1) = 3 \times U(n) + 4$.

Exercice IV : (40 points)

On considère un tableau A [ ] qui contient 50 entiers. Ecrire les fonctions suivantes :

1. *lire_Tab(...)* qui permet de saisir le tableau A [ ].
2. *Min_Tab(...)* qui prend le tableau A [ ] comme paramètre et renvoie le minimum des éléments de A [ ].
3. *Repetition_min(...)* qui prend le tableau A [ ] comme paramètre et retourne le nombre de répétition de l'élément minimum de A[], penser à utiliser la fonction Min_Tab(...) ;
4. *trie_Tab (..)* qui met les éléments du tableau A[] en ordre croissant ;
5. *moy_positif(...)* qui prend comme le tableau A[] comme paramètre et retourne l moyenne des éléments positifs du tableau A[] ;
6. *affiche_tab(...)* qui affiche les éléments du tableau A[] ;

Ecrire un programme en C qui fait appel aux fonctions précédentes et affiche leurs résulta

---

Bon tr

Examen : **Final**

Date : **19 Juin 2017**
Durée : **2 Heures**

Exercice I : (20 points)

Ecrire un programme qui lit un entier positif n et qui calcule et affiche la somme suivante :

$$S_n = \frac{1}{2} + \frac{2}{2^2} + \ldots + \frac{n}{2^n} = \sum_{i=1}^{n} \frac{i}{2^i}$$

Exercice II : (20 points)

Un nombre (de 4-chiffres) est dit Armstrong s'il est égal la somme de ses chiffres élevés à la puissance 4.
Exemple : le nombre 1634 (de 4-chiffres) est Armstrong : $1634 = 1^4 + 6^4 + 3^4 + 4^4$.

Ecrire un programme qui détermine et affiche tous les nombres Armstrong de 4 chiffres

Exercice III : (30 points)

Un nombre N est dit parfait s'il est égal la somme de ses diviseurs différents de N. Ecrire une fonction *parfait (...)* qui prend un entier n comme paramètre d'entrée et qui retourne **true** (ou **1**) si n est un nombre parfait , et sinon retourne **false** (ou **0**) .
Ecrire un programme qui affiche tous les nombres parfaits entre 1000 et 10000. (Note : le programme doit appeler la fonction *parfait(..)* plusieurs fois.)
Exemples des nombres parfaits:

(i) diviseurs de 6 sont 1 , 2 , 3 et $6 = 1 + 2 + 3$
(ii) diviseurs de 28 sont 1 ,2 , 4 , 7, 14 et $28 = 1 + 2 + 4 + 7 + 14$

Exercice IV : (30 points)

On considère un tableau A [ ] qui contient 10 entiers. Ecrire les fonctions suivantes :
lire_Tab(...) qui permet de lire le tableau A [ ].
impair_Tab(...) qui prend un tableau A [ ] comme paramètre et retourne le nombre des entiers impairs dans A [ ].
Remplace_Max(...) qui prend comme paramètres le tableau A [ ] et remplace chaque entier A[i] par le maximum des A[j}, $\forall j , j \leq i$.
Exemple :
A[ ] = 1 , 3 , 5 , 2 , 15 , 51 , 22 , 8 , 11 , 23 devient A[ ] = 1 , 3 , 5 , 5 , 15 , 51 , 51 , 51 , 51 , 51 .
Print_Array(..) qui prend A[ ] comme paramètre d'entrée et affiche les éléments de A[ ] .

Ecrire un programme qui teste les fonctions précédentes.

Bon travail

2016 - 2017 1ère session I1101

EXII :

```
#include<stdio.h>
void main ( ){
int i , a, b, c, d;
for (i = 1000; i<10000; i++)
{k = i;
 a = i/1000;
i = i%1000;
 b = i/100;
i = i%100;
 c = i/10;
i = i%10;
 d = i ;}
if(k == (a*a*a*a)+ (b*b*b*b)
   + (c*c*c*c)+(d*d*d*d) )
printf ("%d est armstrong", i);
}
```

Cours: I1101
Examen : **Final**

Date : **18 /9/ 2017**
Durée : **2 Heures**

Exercice I : (25 points)

Soit la fonction suivante:

```
int fonct(int a, int b)
{
   int t = a, u = b, v = 0 ;
   while (u > 1)
   {
      if (u % 2  != 0)
          v = v + t ;
      t = t + t ;
      u = u / 2 ;
   }
   return t+v ;
}
```

1. Exécute cette fonction pour a=12 et b=21.
2. Que fait cette fonction en général?

Exercice II (30 points)

Ecrire un programme en C qui:

1) lit un tableau A de 20 entiers et un entier x,
2) Calcule et affiche le nombre de répétition de l'entier x dans le tableau A,
3) teste si les éléments du tableau A sont tous distincts ou non,

Exercice III (45 points)

Soit *A*[ ] un tableau dans lequel on a stocké les températures (integers) de 90 jours prises d'un certain région pour la période ***1/1/2017*** à ***31/3/2017*** (A[0] signifie la température de 1/1/2017 , A[1] celle de 2/1/2017 …. A[89] pour la température du dernier jour 31/3/2017) .

Ecrire les fonctions suivantes:

1- *lire_temp(..)* qui lit le tableau *A* [ ] de températures pour le 90 jours.
2- *max_min_temp(..)* qui ***affiche*** le maximum et le minimum des températures dans cette période,
3- *dessous_zero(..)* qui retourne le nombre des jours ayant une température négative,
4- *dessous_moyen(..)* qui retourne le nombre des jours ayant une température au-dessous de leur moyenne dans cette période,

Ecrire un programme qui fait appel aux fonctions précédentes pour afficher les résultats demandés.

Bon travail

Université Libanaise
Faculté des Sciences III
Tripoli – Liban

Cours: **I1101**
Examen : **Final**

Date : **23 Juin 2016**
Durée : **2 Heures**

Exercice I : (30 points)

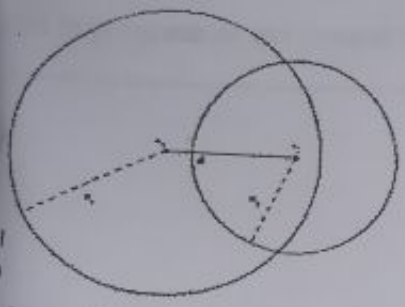
Un point P dans un plan orthonormé est représenté par ces coordonnes Xp et Yp de type float. Un cercle est représenté par les coordonnées de son center O et par son rayon de type float.

Ecrire les fonctions suivant :

1. *distance(...)* qui prend en entré les coordonnes des deux points A et B et qui retourne la distance entre ces deux points.
2. *dans_cercle(...)* qui prend en entré un point A et un cercle C et qui retourne ***vrai*** si le point A est à l'intérieur du cercle C (ou sur sa circonférence) et ***faux*** sinon. Cette fonction utilise la fonction *distance(...)*.
3. *position_cercles(...)* qui prend en entré deux cercles C1 et C2 et qui affiche un message décrivant la position de C1 par rapport à C2 (voir les 5 schémas suivants)

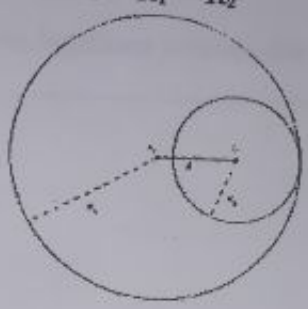
Ecrire le programme qui permet de tester les fonctions précédentes.

$R_1 - R_2 < d < R_1 + R_2$



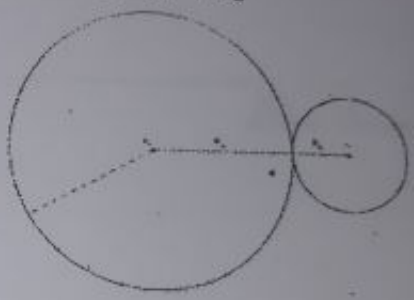
cercles sécants

$d = R_1 - R_2$



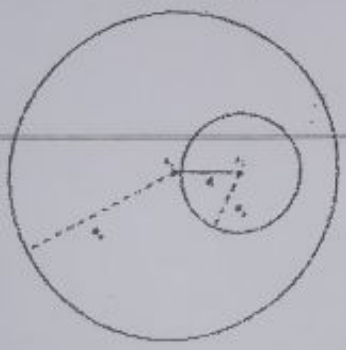
cercles tangents intérieurement

$d = R_1 + R_2$



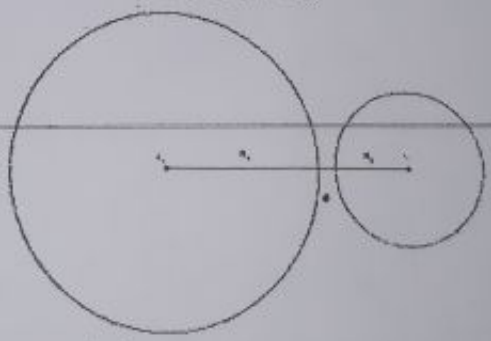
cercles tangents extérieurement

$d < R_1 - R_2$



Internes

$d > R_1 + R_2$



Externes

Exercice II : (30 points)

Un polynôme $P(x) = a_0 + a_1x + a_2x^2 + .. + a_nx^n$ de degré n utilise un tableau coef [ ] pour enregistrer les coefficients a_i. Dans le reste de l'exercice en considère que tous les polynômes sont de degré 10. Ecrire les fonctions suivantes :

1. *lire_Coef(...)* qui permet de lire les coefficients du polynôme *P(x)*.

2. valeur_Poly(…) qui prend en entré le tableau coef [ ] représentant le polynôme $P(x)$ et un réel x_0 et qui calcul et retourne $P(x_0)$.
3. ajout_Poly(…) qui prend en entré trois tableaux, deux tableaux représentants $P(x)$ et $Q(x)$ et qui construit le troisième tableau qui correspond au polynôme $S(x) = P(x)+Q(x)$.
4. affiche_Array(…) qui permet d'afficher les éléments du tableau coef[ ].

Ecrire le programme qui permet de tester les fonctions précédentes.

Exercice III : (40 points)

On considère un tableau A [ ] qui contient 20 entiers. Ecrire les fonctions suivantes :

1. *lire_Tab(...)* qui permet de lire le tableau A [ ].
2. *paire_Tab(...)* qui prend un tableau A [ ] comme paramètre et renvoie le nombre de nombres pairs dans A [ ].
3. *split_Tab(...)* qui prend comme paramètres le tableau A [ ] et deux tableaux *paire [ ]* et *impaire [ ]* et divise les éléments du tableau A [ ] en nombres pairs et impairs. Le tableau paire [ ] contient les éléments paires de A [ ] et le tableau impair [ ] contient les éléments impairs.
4. *merge_Tab (..)* qui prend comme arguments les tableaux paire [ ], impair [ ] et le tableau C [20] et fusionne les tableaux paire [ ] et impaire [ ] dans C [ ].
   Exemple: paire [ ] = **[2, -4, 66, 40, 28]** et impaires [ ] = [-73, 3, 51], puis C [ ] = **[2, -4, 66, 40, 28, -73, 3,51]**

Ecrire le programme qui permet de tester les fonctions précédentes.

Bon travail

Université Libanaise
Faculté des Sciences III
Tripoli – Liban

Cours: **I1101**
Examen : **Session 2**

Date : **24 Septembre 2016**
Durée : **2 Heures**

Exercice I : (20 points)

Considérons la fonction suivante :

```
int f(int a, int b)
{
    int t = a, u = b, v = 0 ;
    while (u > 1)
    {
        if (u % 2 != 0)
            v = v + t ;
        t = t + t ;
        u = u / 2 ;
    }
    return t+v ;
}
```

1. Exécuter cette fonction pour a=12 et b= 21.
2. Déduire ce que fait la fonction en général ?

Exercice II : (30 points)

-Ecrire en C la fonction qui permet de compter le nombre de diviseurs d'un entier x (1 et x ne sont pas considérés comme des diviseurs). Exemple: si x = 30, la fonction retour 6 (les diviseurs sont: 2, 3, 5, 6, 10, 15).

- Ecrire le programme principal qui permet de lire un tableau de 20 entiers, puis affiche l'entier (s) qui a le nombre maximal de diviseurs (si plusieurs entiers vérifier cette règle, il faut les afficher **tous**).

Exercice III : (50 points)

Un employé utilise un tableau (A [7] d'entiers) pour stocker le nombre hebdomadaire d'heures de travail (du lundi au dimanche). A [0] pour lundi, A [1] pour mardi .... A [6] pour dimanche. Le travail en samedi et dimanche est considéré comme heure supplémentaire et sera payé avec 50% en supplément du taux normal. Ecrire les fonctions:

1- read_Input (..) qui lit le tableau A.
2- calculate_salary (..) qui renvoie le salaire hebdomadaire de l'employé. Il est connu que le taux normal de payement est de 6 $ / heure.
3- store_data (..) qui appelle 20 fois les fonctions read_Input (..) et calculate_salary (..) et stocke les salaires des 20 employés dans le tableau «salaire []».
4- sort_salary (..) qui trie le tableau «salaire [] » dans un ordre croissant.

Écrire un programme complet qui appelle les fonctions ci-dessus.

Bon travail