LIF 1 – Lundi 17 décembre 2012
Durée 2h

Les documents, calculatrice, téléphone portable et autres appareils de communication sont interdits
La qualité de l'écriture et de la présentation sera prise en compte dans la note finale.
Vous veillerez à respecter les notations et les règles d'écriture des algorithmes vues en cours et en TD.
Le barème est donné à titre indicatif.

Partie A : Questions de cours (4 points)

Soit le programme C/C++ suivant :

```
#include <iostream>
using namespace std;

void cec (int n1, char c1,int n2)
{ int i;
  for (i=0 ; i<n1 ; i++) { cout<<c1; }
  for (i=0 ; i<n2 ; i++) { cout<<" "; }
  for (i=0 ; i<n1 ; i++) { cout<<c1;}
  cout<<endl;
}

void mystere (int n, char c)
{ int i;
  for (i=0;i<n/2;i++)  { cec(i+1,c,(n+1) - (2 *(i+1))); }
  for (i=n/2;i>=0;i--) { cec(i+1,c,(n+1) - (2 *(i+1))); }
}

int main (void)
{
    mystere (7,'*');
    return 0;
}
```

Donnez le résultat de l'affichage produit lors de l'exécution de ce programme.

Partie B : Algorithmique (10 points)

1 – Nombres heureux (5 pts)

Un nombre heureux est un nombre entier positif qui, lorsqu'on ajoute les carrés de chacun de ses chiffres, puis les carrés des chiffres de ce résultat et ainsi de suite **jusqu'à l'obtention d'un nombre à un seul chiffre**, donne 1 pour résultat. Exemple 7 est heureux, puisque : $7^2 = 49$ et $4^2 + 9^2 = 97$ et $9^2 + 7^2 = 130$ et $1^2 + 3^2 + 0^2 = 10$ et $1^2 + 0^2 = 1$ (on est arrivée à un nombre d'un seul chiffre = 1, donc 7 est heureux)

- écrire l'algorithme d'une fonction `somme_carre` permettant de calculer la somme des carrés des chiffres composant un nombre entier passé en paramètres. Exemple `somme_carre` (49) → 97
- écrire l'algorithme d'une fonction booléenne `heureux?` qui retourne vrai si un entier n passé en paramètre est heureux, faux sinon
- écrire le programme principal permettant d'afficher tous les nombres heureux inférieurs à 100

2- Anagrammes (5 pts)

Deux mots sont anagrammes s'ils sont composés des mêmes lettes.
Exemples : "parisien" et "aspirine" / "villeurbanne" et "invulnerable" / "soigneur" et "guerison".

- écrire l'algorithme d'une fonction booléenne `anagramme?` qui retourne vrai si deux mots passés en paramètres sont anagrammes l'un de l'autre et faux sinon.
  Indication : Pour résoudre ce problème on procédera de la manière suivante : on associera à la deuxième chaine un tableau de booléens initialisé à "false". Pour chacun des caractères de la première chaine, on recherchera s'il existe dans la seconde et on marquera à vrai le booléen correspondant s'il n'est pas déjà à "true". Si à la fin du parcours de la chaine 1, le tableau de booléens est entièrement rempli de "true" les deux mots seront des anagrammes.

- Ecrire l'algorithme du programme principal permettant de saisir 2 mots et de déterminer si ces mots sont deux anagrammes.

Partie C : Langage C (6 points)

1- Définir une structure `temps` permettant de stocker le nombre d'heures, de minutes et de secondes d'un temps.

2- En utilisant la structure précédemment définie, écrire en langage C/C++ les sous-programmes suivants :
   - `temps_secondes` : convertit un temps donné en nombre de secondes.
     Exemple 1h 10m 30s correspond à 4230 secondes
   - `secondes_temps` : convertit un nombre de secondes en temps.
     Exemple 610 secondes correspondent à 0h 10m 10s
   - `somme_temps` : additionne deux temps donnés.
     Exemple 2h 30m 40s + 1h 40m 30s = 4h 11m 10s. Attention aux retenues !
   - `difference_temps` : différence de deux temps donnés, le premier étant supérieur au second.
     Exemple 2h 30m 40s - 1h 40m 30s = 0h 50m 10s

3- Ecrire en langage C/C++ le programme principal permettant :
   - de saisir 2 temps en secondes (on recommencera la saisie tant que le deuxième temps n'est pas strictement inférieur au premier)
   - de transformer ces deux temps en heures / minutes / secondes,
   - de les additionner, de les soustraire, et d'afficher les deux résultats en secondes.