Travail dirigé No. 1
Rédaction d'algorithme

Objectifs : Apprendre à rédiger correctement un algorithme
Durée : 1 semaine
Remise du travail : Avant 23h30, le 18 septembre 2014.
Travail préparatoire : Lecture des exercices.
Documents à remettre : Les algorithmes complétés.

Pour chaque exercice :

- Décrivez l'algorithme de manière générale, en français, sans ternir compte des contraintes du langage simple décrit ci-dessous. Dans cette description, identifiez clairement :
  - ce que l'ordinateur doit afficher à l'usager,
  - ce que l'ordinateur doit lire de l'usager,
  - où sont les conditions,
  - où sont les répétitions (qui n'ont pas à être sous forme « TANT QUE »).
- Écrivez une version raffinée de l'algorithme exprimée uniquement à l'aide des opérations élémentaires suivantes :
  - Lire
  - Afficher
  - = (affecter)
  - TANT QUE *condition* FAIRE …
  - SI *condition* ALORS …
    SINON …
  - Comparaisons : <, >, ≤, ≥, ==, ≠
  - Opérateurs booléens : et, ou, pas/non
  - Opérateurs arithmétiques :
  - + (additionner)
  - − (soustraire)
  - * (multiplier)
  - / (diviser)
  - % (reste ou modulo)

  Les conditions et répétitions doivent être correctement indentées. Les opérations mathématiques suivantes sont permises : sinus, cosinus, valeur absolue, racine carré. Les différents éléments d'une suite ou d'une chaine de caractères sont référés avec les crochets, ainsi, *valeurs[n]* est le $n^{\text{ième}}$ élément de la suite. « longueur de » permet de savoir combien de valeurs/caractères se trouvent dans une suite/chaine.

Exemple : Écrire un algorithme qui vérifie si un nombre entré par l'usager est premier ou non.
Une solution possible :

Demander le nombre à l'usager (affichage). Lire le nombre *n* de l'usager.
Pour chaque entier (une répétition) entre 2 et la racine carrée de *n*, vérifier (une condition) est-ce que cet entier divise *n*.
Si (une condition) aucun des entiers testés ne divise *n*, afficher que le nombre est premier, sinon afficher qu'il ne l'est pas.

Algorithme raffiné :

```
Afficher « Entrer le nombre à vérifier : »
Lire n
i = 2
a trouvé un diviseur = faux
TANT QUE i ≤ √n FAIRE
        SI n % i == 0 ALORS
                a trouvé un diviseur = vrai
        i = i + 1
SI a trouvé un diviseur ALORS
        Afficher « Le nombre n'est pas premier »
SINON
        Afficher « Le nombre est premier »
```

1- Demander et lire les coordonnés de trois points (*A*, *B* et *C*) dans le plan cartésien. Afficher la norme des vecteurs $\overrightarrow{AB}$ et $\overrightarrow{AC}$. Trouver et afficher les coordonnées du point *D* qui forme le parallélogramme *ABCD*. Noter que le langage ne sait pas faire des opérations sur des vecteurs.

2- Demander et lire une suite de 50 nombres entiers non décroissants, pouvant contenir des occurrences multiples d'une valeur, et afficher la suite sans ces occurrences multiples. Par exemple, si la suite saisie est -5 -5 -5 -1 0 1 1 2 3 4 4, alors la suite affichée sera -5 -1 0 1 2 3 4. Vous n'avez pas à vérifier que la suite est non décroissante.

3- Demander et lire deux phrases et une *lettre*. Utilisant **une seule** structure de répétition, déterminer laquelle des deux phrases contient le plus grand nombre d'occurrences de cette *lettre*.

4- Demander et lire toutes les valeurs entières entrées au clavier par l'usager jusqu'à la lecture du nombre 0. Une fois le nombre 0 lu, on doit afficher la somme de toutes les valeurs positives et la somme de toutes les valeurs négatives lues précédemment.

5- Demander et lire 3 nombres. Vérifier si les 3 nombres satisfont le théorème de Pythagore. Si ce n'est pas le cas, redemander et relire les 3 nombres tant que le théorème n'est pas respecté.

6- Demander à l'usager d'entrer des nombres entiers appartenant à l'intervalle [0, 99]. Lire ces nombres jusqu'à la lecture d'un nombre n'appartenant pas à cet intervalle. Pour chaque nombre lu, le message « ce nombre contient le chiffre 3 » est affiché si c'est le cas (p.ex. 3, 13, 23, 33, 35, etc.), et à la fin, la moyenne de ces nombres (qui contiennent le chiffre 3) doit être affichée.

7- La suite de Conway est une suite mathématique inventée en 1986 par le mathématicien John Horton Conway. Le premier terme de la suite de Conway est posé comme égal à 1. Chaque terme de la suite se construit en annonçant le terme précédent, c'est-à-dire en indiquant combien de fois chacun de ses chiffres se répète.

Concrètement :

$X_0 = 1$.

Ce terme comporte juste un « 1 ». Par conséquent, le terme suivant est :

$X_1 = 11$.

Celui-ci est composé de deux « 1 » :

$X_2 = 21$.

En poursuivant le procédé :

$X_3 = 1211$, $X_4 = 111221$, $X_5 = 312211$, $X_6 = 13112221$ et ainsi de suite.

Donner les 100 premiers éléments de la suite de Conway. Pour fin de simplification, considérer chaque terme de la suite comme étant une chaine de caractères. Remarquez que les seules valeurs possibles dans la suite sont '1', '2' et '3'.