TD d'algorithmique élémentaire
Enoncés

Exercice 1 :

Ecrire un algorithme permettant de calculer factoriel n :

n ! = n(n-1)(n-2)….3x2x1

Utiliser d'abord un tantque puis un pour.

Exercice 2 :

Ecrire un algorithme permettant d'afficher tous les diviseurs d'un entier positif saisi.

Entrer un entier positif : 15
Les diviseurs sont : 1 3 5 15

Exercice 3 :

Ecrire un algorithme permettant de tester si un entier positif saisi est premier ou non. Rappel. Un entier est dit premier s'il n'est divisible que par 1 et lui-même.

Entrer un entier positif : 15
L'entier 15 n'est pas premier

Entrer un entier positif : 11
L'entier 11 est un nombre premier

Ecrire le programme avec un pour, puis un tantque.

Exercice 4 :

Ecrire un programme permettant de saisir une date (3 entiers, représentant le jour, le mois et l'année) et de vérifier si cette date est cohérente. En effet, le 32 janvier 2000 n'existe pas, ni le 30 février 2001.

Rappels :
Les mois de janvier, mars, mai, juillet, août, octobre et décembre ont 31 jours.
Les mois d'avril, juin, septembre et novembre ont 30 jours
Le mois de février a 28 jours si l'année n'est pas bissextile, sinon il a 29 jours.

Une année est bissextile si elle est divisible par 4 mais pas par 100. L'année est aussi bissextile si elle est divisible par 400.

Exercice 5 :

Ecrire un algorithme permettant d'afficher tous les nombres parfaits compris entre 1 et *n*, avec *n* saisi par l'utilisateur.
Rappel : un nombre est dit parfait lorsqu'il vaut la somme de tous ses diviseurs (lui-même exclu). Exemple : 6 est un nombre parfait, car 6 = 1+2+3

On décomposera l'algorithme en adoptant une approche descendante.

Exercice 6 :

Ecrire un algorithme de jeu de devinette. Le programme choisit aléatoirement un nombre compris entre 1 et *n*, *n* saisi par l'utilisateur. Le nombre que la machine choisi n'est évidemment pas affiché, mais doit être deviné par le joueur. Pour cela, le programme demande un nombre (qui doit être compris entre 1 et *n*), puis affiche si le nombre saisi par l'utilisateur est plus grand ou plus petit que celui qui a été choisi. Le jeu s'arrête lorsque le joueur a découvert le bon nombre. Le score est alors affiché : le nombre d'essais qu'il aura fallu pour trouver la bonne valeur. Puis, on recommence. La machine choisit un nouveau nombre entre 1 et *n*, etc. Le programme s'arrête lorsque l'utilisateur n'a plus envie de jouer. Il indique ceci en saisissant un nombre négatif. (Utiliser une méthode d'analyse descendante).

Exemple :

Entrer la borne supérieure : 100
J'ai choisi un nombre entre 1 et 100…à vous de le deviner.
Essai numéro 1 (entrer un nombre) : 50
Le nombre choisi est plus grand !
Essai numéro 2 (entrer un nombre) : 83
Le nombre choisi est plus petit !
Essai numéro 3 (entrer un nombre) : 52
Vous avez gagné (score : 3 essais), le nombre choisi était bien 52 !
J'ai choisi un nombre entre 1 et 100…à vous de le deviner.
Essai numéro 1 (entrer un nombre) : 50
Le nombre choisi est plus petit !
Essai numéro 2 (entrer un nombre) : -1
Plus envie de jouer… à bientôt.

On suppose qu'il existe une fonction mathématique *random()* permettant de choisir un nombre réel aléatoire compris dans l'intervalle [0,1[. Par exemple, le programme suivant affiche 10 nombres aléatoires compris entre 1 et 99 :

```
Pour i de 1 à 10 faire
        Afficher entier(100.0*random())
        A la ligne
finfaire
```

Exercice 7 :

Reprendre l'exercice du TP : saisir une date et afficher si elle est valide. Cette fois, on n'utilisera qu'un seul si alors sinon en fin d'algorithme pour afficher le résultat. Indication : utiliser des variables booléennes et des affectations d'expressions booléennes.

Exercice 8 :

Ecrire un algorithme permettant d'afficher une pyramide d'étoiles :

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

Pour cela l'utilisateur saisit la hauteur de la pyramide (dans l'exemple, cette hauteur vaut 6). On utilisera une analyse descendante pour décomposer le problème.