TP 2 *Boucles, tableaux, chaines de caractères*

1. Table de multiplication

Écrire le programme qui demande un nombre x à l'utilisateur, puis affiche la table de multiplication associée.

2. Damier

Afficher un damier de N lignes et N colonnes (N saisi par l'utilisateur), en affichant alternativement les caractères '#' et ' '.

3. Moyenne de notes

Écrire le programme de calcul de la moyenne d'une suite de notes en terminant la saisie par –1. À chaque saisie on vérifiera que la note est comprise dans l'intervalle 0 à 20.

4. Nombre parfait

Un nombre entier naturel *n* est dit parfait s'il est égal à la moitié de la somme de ses diviseurs (1 et *n* inclus). On peut vérifier que les quatre premiers nombres parfaits sont 6, 28, 496 et 8128.

a) Écrire un programme qui détermine si un nombre donné est parfait.

b) Écrire un programme qui affiche tous les nombres parfaits inférieurs à une borne donnée.

5. Liste de nombres

Afficher (dans l'ordre croissant) tous les nombres de 100 à 999 ayant trois chiffres distincts dont la somme vaut 9.

6. Jeu de dé[1]

Deux joueurs lancent chacun un dé à 6 faces. Un joueur qui obtient le plus grand résultat (strict) marque un point. Le jeu s'arrête quand l'un des deux joueurs a obtenu un total de 5 points.

Écrire un programme simulant ce jeu au moyen d'un affichage approprié.

7. Nombre inconnu

Écrire un programme permettant de jouer au nombre inconnu. Un nombre entier aléatoire entre 1 et 100 est stocké en mémoire et l'utilisateur fait des propositions de nombres. À chaque fois, on lui indique si le nombre qu'il a proposé est trop petit ou trop grand. Lorsque le nombre inconnu est trouvé, le programme afficher le nombre de propositions faites.

8. Le plus proche

Écrire un programme qui demande dans un premier temps à l'utilisateur de saisir N nombres réels (N étant une constante). L'utilisateur saisira ensuite un nombre de référence, puis le programme affichera quel nombre, parmi les N saisis préalablement, est le plus proche de la référence.

9. Tableau aléatoire

Écrire un programme qui affiche une suite aléatoire de valeurs entières dont le nombre et les bornes sont spécifiées en constantes.

---

[1] À propos de la génération de nombres aléatoires :

- `rand()` de la bibliothèque `cstdlib` permet de générer un nombre (pseudo-)aléatoire entier, donc `rand()%n` génère un nombre entre 0 et n-1.
- Afin d'initialiser la suite aléatoire et ainsi obtenir des exécutions différentes du même programme, il convient d'exécuter au préalable l'instruction `srand(time(NULL))` (une seule fois, par exemple en début de programme). Cela nécessite l'utilisation de la bibliothèque `ctime` en plus de `cstdlib`.

10. Palindrome

a) Écrire un programme qui demande à l'utilisateur un mot et qui teste si le mot saisi est un palindrome.

Exemple : `LAVAL`, `ELLE` sont des palindromes.

b) Étendre le programme afin de tester les palindromes de phrases (c'est-à-dire en ignorant les espaces).

11. Anagrammes

Écrire un programme qui demande de saisir deux chaines de caractères, puis qui affiche si la deuxième chaine est une anagramme de la première (i.e. formée des mêmes lettres exactement).

Par exemple, CHIEN est une anagramme de NICHE, mais LIVRER n'est pas une anagramme de VRILLE.

12. Histogrammes

Écrire un programme permettant de saisir une suite de notes comprises entre 0 et 20 (la saisie s'arrête pour toute valeur hors des bornes), tout en calculant le nombre d'occurrences de chacune des notes. (On utilisera un tableau Nb tel que Nb[i] représente le nombre d'occurrences de la note i-1).

Afficher ensuite le tableau Nb sous forme d'un histogramme horizontal. Par exemple :

1	3	0	5	
0	1	2	3	…

sera représenté sous la forme :

```
0 : *
1 : ***
2 :
3 : *****
```

Modifier le programme pour afficher un histogramme vertical.

13. Pendu

On demande à l'utilisateur de saisir un mot `sol` (la solution du jeu), puis on part d'une chaine ne contenant que des caractères '_'. On demande alors à l'utilisateur de saisir une lettre, on dévoile alors les lettres correspondantes dans la solution.

Itérer le processus jusqu'à ce que le mot soit entièrement découvert. Indiquer alors le nombre d'essais réalisés.

Voici un exemple de déroulement du jeu (sans effacer la solution).

```
Entrez votre mot mystère: BONJOUR
Mot mystère: _______
Entrez une lettre: J
Après 1 essai, voici le mot mystère: ___J___
Entrez une lettre: E
Après 2 essais, voici le mot mystère: ___J___
Entrez une lettre: O
Après 3 essais, voici le mot mystère: _O_JO__
...
```

Note : Pour effacer le mot mystère après l'avoir saisi, il faut utiliser l'instruction `std::system("clear")` de la librairie cstdlib qui permet d'effacer l'écran