Exercice 1

Thèmes abordés : algorithmique – chaînes de caractères – complexité.

Cet exercice propose l'étude d'un algorithme de détection de palindrome. On rappelle les définitions suivantes :

Définition 1 : un mot est un **palindrome** s'il peut se lire aussi bien dans les deux sens, par exemple le mot « kayak »

Définition 2 : un **variant de boucle** est une suite de valeurs d'entiers positifs strictement décroissante

On considère la fonction `palindrome1` qui renvoie un booléen et dont le paramètre `mot` est une chaîne de caractères de longueur `n`.

```
1|   Fonction palindrome1(mot) :
2|     Variables : i,j : ENTIER ; p : BOOLEEN
3|     i ← 0
4|     j ← longueur(mot)-1
5|     p ← Vrai
6|     tant que i≤j
7|       Si mot[i] ≠ mot[j]
8|         p ← Faux
9|       FinSi
10|      i ← i+1
11|      j ← j-1
12|    Fin tant que
13|    Renvoie p
```

1. Exécuter ligne après ligne cette fonction avec comme argument la chaîne de caractères "rotor" en recopiant le tableau suivant. L'étape 1 correspond à la première exécution de la boucle. Attention : toutes les colonnes ne sont pas forcément à remplir intégralement.

	Initialisation	Etape 1	Etape 2	Etape 3	Etape 4	…
`i ≤ j`						
`mot[i] ≠ mot[j]`						
`i`	0					
`j`	4					
`p`	Vrai					

2.

a. Combien de comparaisons de caractères sont réalisées pour cet algorithme avec le mot "rotor" ?

b. Combien de comparaisons de caractères sont réalisées pour un mot de longueur `n` dans le pire des cas ?

3. Montrer que la quantité `(j-i)` est un variant de la boucle `tant que`. En déduire que cette boucle se termine.

4. Combien de fois la boucle est-elle exécutée avec le mot "`routeur`" ? Proposer une amélioration de l'algorithme de la fonction `palindrome1` que l'on appellera `palindrome2` permettant d'éviter les tours de boucle inutiles. Justifier votre proposition.