

Exercice 3. Récursivité (4 pts)

Ecrivez en C une fonction **récurive** qui retourne vrai si un mot est un palindrome et faux sinon. Voici la signature de la fonction à implémenter :

```
bool palindrome(int n, char* mot);
```

Pour rappel, on dit qu'un mot est un palindrome s'il est identique qu'on le lise de gauche à droite ou de droite à gauche. Les mots "kayak", "radar", "ressasser" sont des exemples de palindrome. Le mot "rouler" n'en est pas un.

Exercice 4. Piles (4 pts)

On considère une chaîne de caractères avec des parenthèses telle que "((a))((b)))".

Ecrire une fonction **profondeur** en C qui prend en paramètre une chaîne de caractères et qui retourne la profondeur maximale de parenthèses imbriquées. Sur l'exemple précédent, la fonction retourne 4, car b est entouré par 4 parenthèses ouvrantes et fermantes correspondantes. En cas de défaut de correspondance des parenthèses, la fonction retournera -1. Les parenthèses comptées sont indiquées en gras.

Autres exemples :

Input : S = "(a(b)(c)(d(e(f)g)h)i(j(k)l)m)";

Output: 4

Input : S = "(p((q))((s)t))";

Output: 3

Input : S = "";

Output: 0

Input : S = "b)(c)()";

Output: -1

Input : S = "(b)((c)())"

Output: -1

Implémentez la fonction **profondeur** en utilisant une pile de caractères et une variable compteur pour garder trace de la profondeur maximale courante lors du parcours de la chaîne de caractères.

Pour la pile de caractères, on suppose qu'on dispose d'un type **stack** et des fonctions :

```
void stack_create(stack* s);  
void stack_push(stack* s, char x);  
void stack_pop(stack* s, char* px);  
void stack_peek(stack s, char* px);  
bool stack_is_empty(stack s);  
int stack_size(stack s);
```

Exercice 5. Tableaux à deux dimensions (5 pts)

Ecrire une fonction **dilatation** en C qui prend en paramètre un tableau bidimensionnel contenant des entiers, ainsi que ses dimensions m et n . Elle retourne un autre tableau de même type dont la valeur de chaque case est obtenue par le maximum des cases voisines du tableau passé en paramètre. Les cases voisines sont celles au-dessus, à droite, au-dessous, à gauche et elle-même (voir l'exemple ci-dessous). Pour les cases du bord du tableau, il ne faut prendre que les cases « dans le tableau ».

Voici la signature de la fonction **dilatation** :

```
int** dilatation(int m,int n,int** tab);
```

Découpez votre code en fonctions appropriées. Le tableau retourné doit être alloué dans la fonction à l'aide d'appels à la fonction **malloc**. Les lignes de la matrice ne doivent pas forcément être contiguës en mémoire.

Exemple :

The diagram illustrates the dilatation function's logic. It shows a 7x7 input matrix with values. Two callout boxes highlight specific calculations:

- A callout from the cell at row 1, column 7 (value 0) points to a box containing: **Max (1, 0, 1) = 1**
Il faut remplacer 0 par 1
- A callout from the cell at row 4, column 3 (value 4) points to a box containing: **Max (5, 3, 4, 5, 3) = 5**
Il faut remplacer 4 par 5

13	2	6	4	2	1	0
1	5	2	5	3	2	1
2	3	4	11	1	3	2
3	4	5	3	4	5	6
2	3	4	5	4	2	2
1	12	3	4	13	2	9
9	11	2	3	2	1	10