TP : 5 algorithmes importants

Question 1.

Écrire la fonction `int max_array(const int *a, size_t n)` qui renvoie le maximum d'un tableau d'entiers de taille n. Faire une étude rapide de sa complexité.

Remarque.
- Les tailles et les indices des tableaux peuvent être données sous la forme d'entiers `int` ou bien utiliser le type plus adapté `size_t` (qui désigne des entiers non signés).

  Le spécifieur de format pour ce type est `%zu`.
- En C, lorsqu'on passe un tableau en paramètre de fonction, il se décaye en pointeur. Donc ces trois écritures sont strictement équivalentes pour le compilateur :

```c
int f(int a [], size_t n);
int f(int *a, size_t n);
int f(const int *a, size_t n);

```

- On emploie ici `const int*` au lieu de `int a[]` pour indiquer qu'on n'a pas l'intention de modifier le tableau (et que ça ne devrait pas arriver).

Question 2.

Écrire la fonction `double avg_array(const double *a, size_t n)` qui calcule la moyenne arithmétique d'un tableau non vide. Donner sa complexité temporelle.

Question 3.

Écrire la fonction `int search_array(const int *a, int n, int target)` qui renvoie la position de la cible (-1 si on ne la trouve pas). Donner sa complexité temporelle.

Question 4.

Écrire la fonction `void rev_array(int *a, size_t n)` qui inverse l'ordre des éléments d'un tableau (*en place* donc sans utiliser de tableau auxiliaire). Donner sa complexité temporelle.

Question 5.

Écrire la fonction `bool is_sorted_array(const int *a, size_t n)` qui renvoie vrai si le tableau est trié par ordre croissant et faux sinon. Donner sa complexité temporelle.