Travaux Pratique n°4

Exercice 1

Tri par sélection
Écrire un programme C qui implémente l'algorithme de tri par sélection vu en TD. On définit N le nombre d'éléments du tableau à trier comme une constante.

Exercice 2

Tri à bulle
Écrire un programme C qui implémente l'algorithme de tri à bulle vu en TD. On définit N le nombre d'éléments du tableau à trier comme une constante.

Exercice 3

Tri par insertion
Écrire un programme C qui implémente l'algorithme par insertion vu en TD. On définit N le nombre d'éléments du tableau à trier comme une constante.

Exercice 4

Le tri de Shell
Le tri de Shell est une amélioration du tri par insertion en observant deux choses :

(1) Le tri par insertion est efficace si la liste est à peu près triée.

(2) Le tri par insertion est inefficace en moyenne car il ne déplace les valeurs que d'une position par instruction.

Le tri de Shell trie chaque liste d'éléments séparés de h positions chacun avec le tri par insertion. L'algorithme effectue plusieurs fois cette opération en diminuant h jusqu'à $h = 1$ ce qui équivaut à trier tous les éléments ensemble.
Le fait de commencer avec des éléments espacés permet de pallier l'inconvénient **(2)**, tandis que lorsque l'on fait à la fin avec un espacement de 1, ce qui est en fait un tri par insertion ordinaire, on tire parti de l'avantage **(1)**.
Implémenter l'algorithme de trie de Shell en modifiant le programme C du tri par l'insertion. On choisit les pas h dans la séquence $1, 4, 13, 40, 161, ...$, c.à.d. $h_{k+1} = h_k * 3 + 1$. A initialisation, on choisit le pas h maximal tel que $h < N$ et après chaque itération dans laquelle, on fait le tri par l'insertion avec le pas h, on diminue h par $h = h/3$.

Exercice 5

Bonus : Comparaison des méthodes de tri implémentées
On génère aléatoirement un tableau de 30000 entiers et on exécute successivement des méthodes de tri que l'on a implémenté dans les exercices précédents. On mesure le temps d'exécution

de chaque méthode et l'afficher à l'écran. Pour générer le tableau, on utilise les instructions suivantes :

```
srand (time (NULL));
for (k = 0; k < N; k++)
    arr[k] = rand()%R_MAX;
```

où *arr* est le tableau, R_MAX est une constante représentant la valeur maximale que puissent prendre les éléments du tableau.
Pour mesurer le temps (en secondes) d'exécution d'une séquence d'instruction, on utilise la bibliothèque "time.h". On déclare une variable p de type "double" et deux variables "start" et "end" de type "clock_t" :

```
double p;
clock_t start, end;
```

et au début et à la fin de la séquence d'instruction à mesurer le temps d'exécution, on consulte l'horloge de l'ordinateur :

```
start = clock();
//la séquence d'instruction
end=clock();
```

On calcule et affiche la difference en secondes :

```
p=(double) (end - start) / CLK_TCK;
printf("Le temps d'execution  est : %lf secondes\n", p);
```