Contrôle DV5.1

▼ Exercice 1

▼ Exercice 1 A

commande d'execution `gcc -o a.out main.c <racine à tester>`

```
int racineCarree(int n){
    for(int i = 0; i < n; i++) {
        if (i*i == n) {
            return i;
        }
    }
    return -1;
}
```

▼ Exercice 1 B

commande d'execution `gcc -o a.out main.c <taille tableau>`

```
int* racineCarreeTab(int base_tab[], int tabSize) {
    int* computedTab = (int*)malloc(tabSize * sizeof(int

    for (int i = 0; i < tabSize; i++) {
        computedTab[i] = racineCarree(base_tab[i]);
    }

    return computedTab;
}
```

▼ Exercice 2

commande d'execution :

```
gcc -g -pg -o main main.c
./main <taille tableau>
```

```
gprof main gmon.out > temp.txt
cat temp.txt
```

Profiling pour 50 éléments :

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
100.23    115.62   115.62       50     2.31     2.31  racir
  0.00    115.62     0.00        1     0.00   115.62  racir

 %          the percentage of the total running time of the
time        program used by this function.



index % time    self  children    called     name
              115.62    0.00       50/50          racineCarr
[1]     100.0  115.62    0.00       50          racineCarree [
-----------------------------------------------
                0.00  115.62       1/1          main [3]
[2]     100.0    0.00  115.62       1          racineCarreeTa
              115.62    0.00       50/50          racineCarr
-----------------------------------------------
                                                 <spontanec
[3]     100.0    0.00  115.62                 main [3]
                0.00  115.62       1/1          racineCarr
```

Complexité cycomatique et algorithmique de `racineCarree`

Complexité cyclomatique : 2

Complexité algorithmique : o(n)

Complexité cycomatique et algorithmique de `racineCarreeTab`

Complexité cyclomatique :

Complexité algorithmique : o(n²)

Exercice 3

```
int* firstSort(int square_root_tab[], int tab_length, int t
    int* computedTab = (int*)malloc(tab_length * sizeof(int

    for (int i = 0; i < tab_length; i++) {
        if(i%2 == 0) {
            computedTab[i] = square_root_tab[i];
        } else {
            computedTab[i] = square_root_tab[i] * tab_value
        }
    }

    return computedTab;
}

int* secondSort(int square_root_tab[], int tab_length)  {
    int* computedTab = (int*)malloc(tab_length * sizeof(int

    for (int i = 0; i < tab_length; i++) {
        if(i%2 == 0) {
            computedTab[i] = racineCarree(square_root_tab[i
        } else {
            int summ = 0;
            int* result = RacineCarreeTab(square_root_tab,
            for(int j = 0; j<tab_length; j++) {
                summ = summ + result[j];
            }
            computedTab[i] = summ * square_root_tab[i];
        }
    }

    return computedTab;
}

int* TriSpecial(int square_root_tab[], int tab_length){
    int tab_value = 0, summ_error = 0;

    for (int i = 0; i < tab_length; i++) {
        tab_value = tab_value + square_root_tab[i];
        if(square_root_tab[i] < 0) {
            summ_error++;
        }
    }

    if(summ_error % 2 == 0) {
        return firstSort(square_root_tab, tab_length, tab_v
    } else {
        return secondSort(square_root_tab, tab_length);
    }
}
```

Exercice 4

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total           
 time   seconds   seconds    calls   s/call   s/call  name    
100.21     45.52    45.52       50     0.91     0.91  racir
  0.00     45.52     0.00        1     0.00    45.52  Racir
  0.00     45.52     0.00        1     0.00     0.00  TriSp
  0.00     45.52     0.00        1     0.00     0.00  first

 %         the percentage of the total running time of the
time       program used by this function.

index % time    self  children    called     name
               45.52    0.00      50/50          RacineCarr
[1]    100.0   45.52    0.00      50         racineCarree [
-----------------------------------------------
                0.00   45.52       1/1           main [3]
[2]    100.0    0.00   45.52       1         RacineCarreeTa
               45.52    0.00      50/50          racineCarr
-----------------------------------------------
                                                 <spontaneo
```

```
[3]    100.0    0.00   45.52                 main [3]
                0.00   45.52       1/1           RacineCarr
                0.00    0.00       1/1           TriSpecial
-----------------------------------------------
                0.00    0.00       1/1           main [3]
[4]      0.0    0.00    0.00       1         TriSpecial [4]
                0.00    0.00       1/1           firstSort
-----------------------------------------------
                0.00    0.00       1/1           TriSpecial
[5]      0.0    0.00    0.00       1         firstSort [5]
```

Complexité cycomatique et algorithmique de `firstSort`

Complexité cyclomatique : 2

Complexité algorithmique : o(n)

Complexité cycomatique et algorithmique de `secondSort`

Complexité cyclomatique : 3

Complexité algorithmique : $o(n^3)$

Complexité cycomatique et algorithmique de `triSpecial`

Complexité cyclomatique : 3

Complexité algorithmique : $o(n^3)$

Soit une complexité cyclomatique **totale** de 8.

▼ Bonus - upgrade code

```
//modification de racinecarree

int racineCarree(int n){
    for(int i = 0; i < n; i++) {
        if (i * i > n) {
            break;
        }
        if (i*i == n) {
            return i;
        }
    }
    return -1;
}

//Complexité = O(log(n))
```

```
BIZAREMENT AVEC UN CODE OPTI C'EST PLUS RAPIDE HEIN

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
100.21      0.71     0.71    10000     0.07     0.07  racin
  0.00      0.71     0.00        1     0.00   711.47  Racin




index % time    self  children    called     name
                0.71    0.00   10000/10000       RacineCarr
[1]    100.0    0.71    0.00   10000         racineCarree [
-----------------------------------------------
                0.00    0.71       1/1           main [3]
[2]    100.0    0.00    0.71       1         RacineCarreeTa
                0.71    0.00   10000/10000       racineCarr
-----------------------------------------------
                                                 <spontaneo
[3]    100.0    0.00    0.71                 main [3]
                0.00    0.71       1/1           RacineCarr
-----------------------------------------------
```