hw10.cpp
環境:使用 vscode,利用 g++ -O2 進行編譯,使用 hw10.txt 15 萬筆測資
運行結果:

```
PS E:\C-projects\mycode\DShw\hw10> ./hw10
use leftmost and longer first execution time = 0.016000
```

```
PS E:\C-projects\mycode\DShw\hw10> ./hw10
use leftmost and shorter first execution time = 0.015000
```

```
PS E:\C-projects\mycode\DShw\hw10> ./hw10
use leftmost and original execution time = 0.015000
```

```
PS E:\C-projects\mycode\DShw\hw10> ./hw10
use median of three and longer first execution time = 0.011000
```

```
PS E:\C-projects\mycode\DShw\hw10> ./hw10
use median of three and shorter first execution time = 0.010000
```

```
PS E:\C-projects\mycode\DShw\hw10> ./hw10
use median of three and original execution time = 0.010000
```

	Longer first	Shorter first	original
leftmost	0.016	0.015	0.015
median of three	0.011	0.010	0.010

先來說 tail recursion optimization 的影響:
先進行 longer sublist 的時間會需要較長一點點，因為開啟 Tail recursion optimization 後，可以將 funtion 的最後一句的 recursion 進行最佳化，可以變成跟 iteration 差不多，可以不用因為執行到該 function 又為他建立一個新的 stack frame，所以對先進行 longer sublist 來說，優化到的是比較短的部分，而對 shorter sublist 來說，優化到的是比較長的部分，所以先進行 longer sublist 會花較多的時間。

leftmost vs median of three:
在 median of three 的表現是較好的，在我的程式中，會先對 left,right,(left+right)/2 進行排序並且找出他的中位數，再來使中位數跟[left+1]交換之後做 leftmost，這樣的好處是可以比較有機會避免掉左右失衡的情況，跟比較有機會避免掉 worst case，並且因為一開始有進行排序了，所以原來的 left 跟 right 就不用再做了，用 left++跟 right—後再來做 leftmost 可以減少時間，而直接做 leftmost 沒有以上的減免。

程式架構:
這邊我利用的不是直接用 quick sort，而是有把 partition 分出來寫並且將 quicksort 跟 partition 分成有做 median 跟沒有做 median 的，跟一個常見的 swap

函式，以及用來看有沒有排好的 print。
在 main 中，先建立一個很大的 array a 並且對其操作，一開始先進行讀檔讀取 hw10.txt，之後利用可以計時的變數來對 quicksort 15 萬筆進行計時，
並且 print 出用了多少時間。在程式的使用中看現在是要看哪個的時間就對該註解進行刪減或打開，如果需要更動看是要 longer first 還是 shorter first 或 original 的話就到函式進行註解的打開以及刪減。

函式:
swap:進行交換

print:print 出 array 中的值，為測試用

Partition:對要 quick sort 的 array 進行分割，使得比 pivot 小的都在左半邊，比 pivot 大的都在右半邊，跟回傳的 pivot 位置

QuickSort:分割完之後先對前半部做 QuickSort，再對後半部進行 QuickSort，recursive 下去使其完成。(如果需要 longer first 還是 shorter first 或 original 的調整話就到裡面進行註解的打開以及刪減)

Partition_medain:對要 quick sort 的 array 進行分割，使得比 pivot 小的都在左半邊，比 pivot 大的都在右半邊，跟回傳的 pivot 位置，再此之前，先做 median of three 來減少 worst case 的發生，減少時間的運行

QuickSort_median: 分割完之後先對前半部做 QuickSort_median，再對後半部進行 QuickSort_median，recursive 下去使其完成。(如果需要 longer first 還是 shorter first 或 original 的調整話就到裡面進行註解的打開以及刪減)