How2Run:

First we can compile our program while it is in the same folder with books.csv:

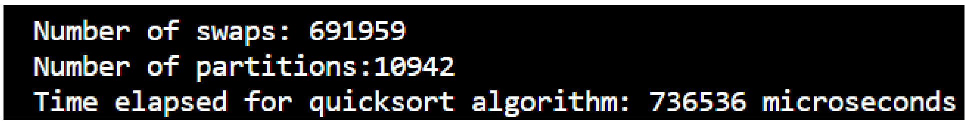
```
g++ -Wall -Werror assg1.cpp -I include -o assg1 -lm
```

We can run it with this:

```
./assg1
```

You can see the sorted array from sorted_books.csv which is in the hw1.zip.

Statistics:



```
Number of swaps: 691959
Number of partitions:10942
Time elapsed for quicksort algorithm: 736536 microseconds
```

About Quicksort Algorithm:

• Best case for the quicksort algorithm is T(n) = T(n/2) + T(n/2) + cn, when every split is even. By master theorem, best case running time is T(n) = ϴ(nlogn). But even there is uneven split like T(n) = T(99n/100) + T(n/100) + cn running time is still O(nlgn) because deepest level is logn/log(100/99) = ϴ(lgn) and work that is done in every level is at most cn

So even there is a unbalanced split, running time is O(nlgn).

• Worst case for quicksort algorithm is when one side of the split is 0 and other is n-1 i.e T(n) = T(n-1) + T(0) + ϴ(n). This generate arithmetic series so solution of this recurrence is T(n) = ϴ(n^2).

• Average case is when there is a good splits and bad splits across the tree. Let's say best and worst cases are back to back. Right after a worst case there is a best case. Worst case costs us ϴ(n) and best case costs us ϴ(n-1) thus ϴ(n) + ϴ(n-1) = ϴ(n). We have 3 arrays which have 0, (n-1)/2 - 1, (n-1)/2 lengths. When there is only best case we have ϴ(n) and two arrays that is (n-1)/2 length. Work that is to be done and costs are nearly same. Thus we can see that when in average case, best cases can absorb worst cases and running time is again O(nlgn).

• Like we said, worst case is when one part of the split is 0 and other is n-1 so T(n) = T(n-1) + T(0) + ϴ(n). We can prevent this by choosing the pivot element randomly in the array.