R-2.8 Illustrate the performance of the selection-sort algorithm on the following input sequence (22, 15, 26, 44, 10, 3, 9, 13, 29, 25).

22, 15, 26, 44, 10, 3, 9, 13, 29, 25. Find the smaller 3

22, 15, 26, 44, 10, 3, 9, 13, 29, 25. Swap 3 and 22

3,15, 26, 44, 10,**22**, 9, 13, 29, 25. Find the smaller 9

3,9, 26, 44, 10,**22**, **15,** 13, 29, 25. Swap 9 and 15

3,9, 26, 44, 10,**22**, **15,** 13, 29, 25. Find smaller 10

3,9,10**,** 44, **26**,**22**, **15,** 13, 29, 25. Swap 10 and 22

3,9,10, 44, **26**,**22**, **15,** 13, 29, 25. Find smaller 13

3,9,10, 13**,** **26**,**22**, **15, 44**, 29, 25. Swap 13 and 44

3,9,10, 13, **26**,**22**, **15, 44**, 29, 25. Find smaller 15

3,9,10, 13, 15,**22**, **26, 44**, 29, 25. Swap 15 and 26

3,9,10, 13, 15,**22**, **26, 44**, 29, 25. Find smaller 22

3,9,10, 13, 15,22, **26, 44**, 29, 25. Swap

3,9,10, 13, 15,22, **26, 44**, 29, 25. Find Smaller 25

3,9,10, 13, 15,22, 25, **44**, 29, 26. Swap 25 and 26

3,9,10, 13, 15,22, 25, **44**, 29, 26. Find smaller 26

3,9,10, 13, 15,22, 25, 26, 29, 44. Swap 26 and 44

3,9,10, 13, 15,22, 25, 26, 29, 44. Find smaller 29

3,9,10, 13, 15,22, 25, 26, 29, 44. Swap 29 and 29

3,9,10, 13, 15,22, 25, 26, 29, 44. End

3,9,10, 13, 15,22, 25, 26, 29, 44. Finally sorted

R-2.9 Illustrate the performance of the insertion-sort algorithm on the input sequence of the previous problem.

22, 15, 26, 44, 10, 3, 9, 13, 29, 25. Shift if less than

15, 22, 26, 44, 10, 3, 9, 13, 29, 25. Shift if less than

15, 22, 26, 44, 10, 3, 9, 13, 29, 25. Shift if less than

15, 22, 26, 44, 10, 3, 9, 13, 29, 25. Shift if less than

15, 22, 26, 44, 10, 3, 9, 13, 29, 25. Shift if less than

15, 22, 26, 44, 10, 3, 9, 13, 29, 25. Shift if less than

10,15, 22, 26, 44, 3, 9, 13, 29, 25. Shift if less than

3,10,15, 22, 26, 44,9, 13, 29, 25. Shift if less than

3,9,10,15, 22, 26, 44,13, 29, 25. Shift if less than

3,9,10,13,15, 22, 26, 44,29, 25. Shift if less than

3,9,10,13,15, 22, 26,29, 44, 25. Shift if less than

3,9,10,13,15, 22, 25,26,29, 44 Shift if less than finally sorted elements

Algorthim isPermation(A,B)

If A.length!=B.length	1
return false;	1
heapsort(A)	nlogn
heapsort(B)	nlogn
for i:=0 to A.length do	n
if A[i]!=B[i]	n
return false	1
return true;	1

Big O of this is O(nlogn+n)

```
const Sorts = require('./HW07-ArraySorter.js');
function isPermutation(A,B){
    if(A.length!==B.length) return false;
    let sort=new Sorts.ArraySorter();
    sort.heapSort(A)
    sort.heapSort(B)
    for(let i=0;i<A.length;i++){
        if(A[i]!==B[i])
          return false;
    }
    return true;
}
let A=[1,4,2]
let B=[4,1,2]
console.log("expected true: "+ isPermutation(A,B))
let C=[5,7,9,2]
```

```
let D=[2,9,7,3]
console.log("expected false: "+ isPermutation(C,D))
```