Taehyun Park 115242740

1.
A. O(n^4)
B. O(1/sqrt(n))
C. O(1)
D. O(1)
E. O(n)

2.

```
public int len() {
        int count=0;
        for(Student s:rawData){     -c1 N
                if(s==null) continue; //Does not count Index that is empty  -c2 N-1
                count++;                  -c3 N-1
        }
        return count;
}
```

T(N) = c1*N + c2(N-1) + c3(N-1)
Big - O ; O(n)

```
public boolean addStudent(Student s) {
        if(find(s) == null){       -c (Trial for 'if' itself is just a single time, but trials of 'find()' is N times)
                for(int i=0;i< rawData.length;i++){    -c1  N
                        if(rawData[i] == null){ //fill in the first index that is empty     -c2 N-1
                                rawData[i] = s;       -c3 X
                                return true;          -c4 X
                        }
                }
        }

        return false;
}
```

T(N) = c1*N + c2(N-1) + c3*X + c4*X
Big - O : O(n)

3.

```
public static double getMinimum(int[] arr) {
        // Fill in here (arr is sorted in either ascending or descending order)
        if(arr[0] < arr[1]) { return arr[0];}    -c1 1
        else {return arr[arr.length-1];}    -c2 1
}
```

The method will run just two lines one time for each, no matter how big is the input.
Big - O : O(1)

4.

```
public static int count(int[] arr, int item) {
        // Return the # of elements that are <= item
        int counting = 0;
        for(int I :arr){.          -c1 N
                if(I<=item){ -c2 N-1
                counting++;   -c3 X
                }
        return counting;
}
```

T(N) : c1*N + c2(N-1) + c3*X
For the best case : c1*N + c2(N-1) + c3*(N-1)
For the worst case : c1*N + c2(N-1)

Average case : c1*N + c2(N-1) + c3*(N-1)/2
Average would make difference with just c3, making it half trials. Because the difference between the best case and the worst case is only trials for c3, and to calculate average, we can just take middle value of those two(Half of trial sum).