COMP 482 Project 3
Due: Monday November 30 2355

Task: You will write a java program which

- reads a 2 sorted lists of integer data of size N from a file,
- stores that data into 2 linear data structures (eg array / java.util.ArrayList),
- finds the average of the 2 "middle" elements in time $O(\lg n)$ using divide and conquer (middle in this case means the two values that would be the n^{th} and $(n+1)^{st}$ if the two lists were merged, but note you cannot merge the list to find the two middle elements, and
- finds the number of inversions in the list that would be created by concatinating the 2 lists in time $O(n)$ using divide and conquer (ie by merging the 2 lists just like we did in class).

Your program must:

- compile using the command 'javac Project3.java',
- run using the command 'java Project3',
- accept input from a file called input3.txt formatted precisely as described below,
- output results exactly as described below, and
- meets the time complexity requirements above.

If your submission fails to do any of these things, expect a score no higher than 5/20.

Your program should:

- be properly formatted (eg using proper indentation)
- be structured reasonably (eg dividing the tasks up into methods), and
- be commented appropriately (eg containg your name at or near the beginning of each file and explaining how/why a method works just above the method).

If your submission fails to do any of these things, expect a score no higher than 15/20.

Input format: The input file, input3.java, will be formatted with the number n on the first line, followed by n sorted integers seperated by whitespace, followed by n sorted integers seperated by whitespace. For example:

```
9
5 13 29 31 45 61 67
77 82
9 11 14 16
19 21 24 27 28
```

Task: First read the data then create and fill two arrays (or ArrayLists) that store the data. For example, with the input sample above you should have two arrays like the following.

5	13	29	31	45	61	67	77	82

9	11	14	16	19	21	24	27	28

Second use a divide and conquer method to find the average of the of the two items that would be in the middle of the array created by merging these two arrays. However note that at this point you cannot merge the arrays because it will take time $O(n)$ instead of $O(\lg n)$. For the sample data you want the average of the 9^{th} and 10^th items if you merged the lists, but to repeat again you can't merge the lists. The trick here is to compare the medians of the two arrays. Using this information you can rule out roughly half of each array and have the same problem as the originally except with half the size. For example, if you compared the fifth items in the arrays you'd note that $45 > 19$. This implies that every item in the first array larger than 45 has 10 items smaller than that it and can be eliminated from consideration. Similarly every item in the second array to the left of 19 has 10 items larger than it and can be eliminated from consideration. You now have the same problem but smaller. You are now looking for the average of the 5^{th} and 6^{th} items if you merged the lists:

5	13	29	31	45

19	21	24	27	28

Third use the technique given in class CountingInversions to merge the arrays and count the inversions.

Output Format: You will output 2 lines. The first line will be the average of the two middle values. The second line will be the number of inversions.

Sample output: For the sample input above:

```
25.5
65
```

Stray Thoughts:
I will be using a recent version of Java (likely the current version Java SE 15, but if Oracle releases a new version I may upgrade).
You'll be submitting only dot-java files (no class files or input files required or wanted).
You are allowed to use any of the standard features, classes, methods in Java. For example, I expect nearly all students will want to use either an array or java.util.ArrayList. You can use as many or as few files as you feel appropriate, but the main method should be located in a file called Project3.java. Otherwise the project won't compile/run with the required commands.
Some IDEs default to placing java files into packages. This will likely cause the commands 'javac Project3.java' and/or 'java Project3' to fail. Either use an IDE that does not place java files into packages OR learn your preferred IDE well enough to avoid this issue OR delete any package lines before submission.
Students often decide to change or modify the format of the input or ouput. Sometimes it makes the project easier for them. Other times a student thinks it is an improved design. You may or may not be right, but don't change the input or output format. Doing so will result in your project getting a low score.
It is likely that many students won't read this far. There is no need to let me know you've read this.
There is are very easy ways to determine answers in time $O(n)$ and $O(n^2)$. You should not use this methods. You should instead use methods which are $O(\lg n)$ and $O(n)$.
I will likely use the sample input file above while grading, but I'm also likely to use other much larger input files (possibly containing millions of data items, but no more than the JVM limit on arrays $\approx 2^{30}$).
I suggest you finish you project several days in advance. This way you have time and opportunity to ask any last questions and verify that what you upload satisfies the requirements.
Your project should be written and understood by you. Helping or receiving help from others is allowed, but significant shared source code indicates that you either did not write/understand what you submitted or you assisted another in submitting code they did not write or understand.