COMP 610 Project 3

Idea: You will be given 2 sorted integer arrays and asked to find the item which would be in position i if the 2 arrays were merged. However, your implementation must run in time $O(\lg n)$.

Note merging the arrays takes time $O(n)$. So solving the problem directly is not allowed. On the other hand, you can use the $O(\lg n)$ time algorithm described in class and for which a short reminder is given here (in the reminder all off by 1 issues will be ignored).

If one of the arrays has size 1 (terminating condition for the recursion) finding the correct answer can be done with a one or two comparisons in constant time.

If both arrays still have size larger than 1, perform the following recursive step. Assume the first (second) array has size s_1 (s_2). Compare the medians of the two arrays. In other words, compare the item in position $s_1/2$ of the first array to the item in position $s_2/2$ of the second array. The smaller median and all items earlier in its array would be in the first $s_1/2 + s_2/2$ positions of the merged array the larger median and all items later in its arrray would be in the last $s_1/2 + s_2/2$ positions of the merged array. Depending upon the position you are seeking, one of these two sets cannot contain the desired item. So half of one of these arrays can be ignored at the next iteration.

Input Format: The input file will be called input3.txt and be in the same directory as the java and class files. The format of input3.txt will be a standard text file containing whitespace (spaces/tabs/newlines) seperated integers. The first line will consist of two integers: N which is the number of items in each array initially and k the position in the hypothetical merged array for which we are searching. The remaining lines will consist of $2N$ integers in the two arrays seperated by white space. The first and second N of these will be sorted.

Output: A single integer which is the item sought.

Examples: If the first array contained 5, 25, 45, 65, 85 and the second array contained 15, 35, 55, 75, 95 and we wanted to find the item that would be in position 7 if it was merged, the input3.txt would contain

```
5 7
5 25 45 65
85 15 35 55
75 95
```

and the output woud be

```
75
```

because the hypothetical sorted array would have 75 in position 7.

Similarly, if input3.txt contained

```
5 3
5 25 45 65
85 15 35 55
75 95
```

the the output would be

```
35
```

Details: The program must be written in Java. The program must compile with the command javac *.java and run with the command java Project3. The program should be reasonably commented, indented, structured. The program should be submitted by placing all files in a directory named after you, zipping this directory and submitting via canvas (ie if the professor was submitting then all files would be placed in a directory called JohnNoga, this directory would be zipped, and uploaded in canvas). Failure to follow these directions will result in serious reductions in your score.

How: You must use a $O(\lg n)$ time algorithm. Note that this means you cannot actually merge the arrays. A submission which does the calculation in time $O(n)$ will receive little credit. The test cases your program will be run against will be much larger than the one above.