Problems on Binary Search - 2

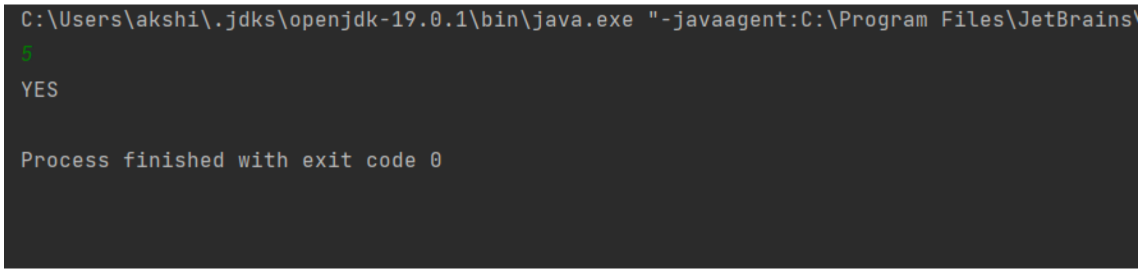
Assignment Solutions

PW SKILLS

Q1. Given a non-negative integer c, decide whether there are two integers a and b such that $a^2 + b^2 = c$.

Code : https://pastebin.com/HHC9du7v

Output:



```
C:\Users\akshi\.jdks\openjdk-19.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\
5
YES

Process finished with exit code 0
```

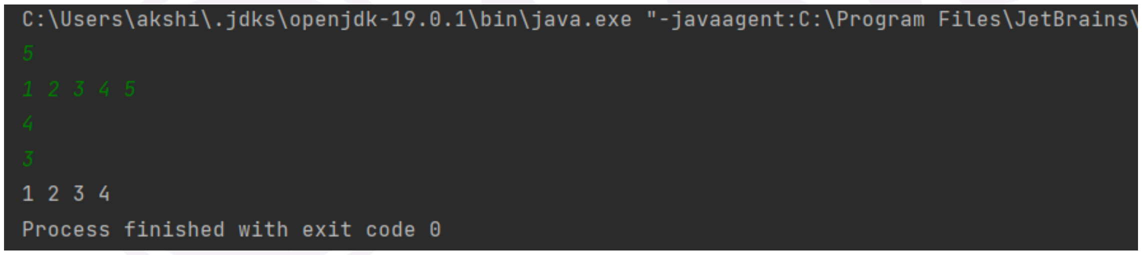
Explanation :

- Here a^2 + b^2 = c, so we will traverse over all the values of a, as we already know c and we are traversing on values of 'a', so we can calculate 'b^2' through that.
- Just check if there is a square root of b^2 or not. If there exist, then there will be a value of 'b' that satisfies the condition.
- We can use recursive binary search to check if that square root exists.

Q2. Given a sorted integer array arr, two integers k and x, return the k closest integers to x in the array. The result should also be sorted in ascending order.

Code : https://pastebin.com/r21Qfz3U

Output:



```
C:\Users\akshi\.jdks\openjdk-19.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\
5
1 2 3 4 5
4
3
1 2 3 4
Process finished with exit code 0
```

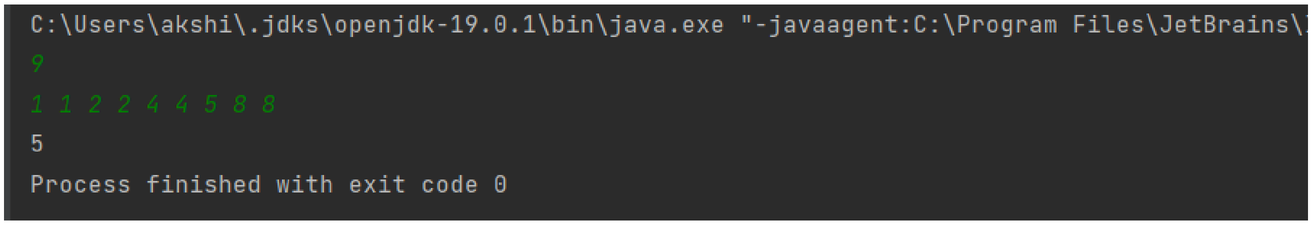
Explanation :

- To find the k closest elements to a particular element, we find the lower bound of that element and then take two pointers and keep moving in opposite directions.
- Keep on adding the answer whenever required.
- Find the lower bound using the binary search algorithm.

Q3. You are given a sorted array consisting of only integers where every element appears exactly twice, except for one element which appears exactly once.
Return the single element that appears only once.

Code : https://pastebin.com/jF9nrgPz

Output:



```
C:\Users\akshi\.jdks\openjdk-19.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\
9
1 1 2 2 4 4 5 8 8
5
Process finished with exit code 0
```

Explanation :

- Let's divide the array into 2 parts-
  - One part is before the single element
  - Second part is after the single element
- Observe that all the elements in the first part have their first occurrence at an even index and second occurrence at an odd index. Further in the second part, the elements have their first occurrence at an odd index and second occurrence at an even index.
- Based on this if we divide the array into groups of 2 elements-[1, 1, 2, 2, 4, 4, 5, 8, 8]→ [1,1], [2,2], [4,4], [5,8], [8,-] Here we can observe that in such a scenario, all the groups formed before the required element have equal values stored in them. But in the latter part, the values are not equal.
- We can apply binary search using this property-
  - For each middle index we find its pair index corresponding index-
    - If the current index is even, we'll get its next index
    - Else we'll get its previous index.
  - Now if the value at the two indices are equal then it means that we are in the first part of the array.
  - Else we are in the second part of the array.
- We want the first element in the second part of the array.