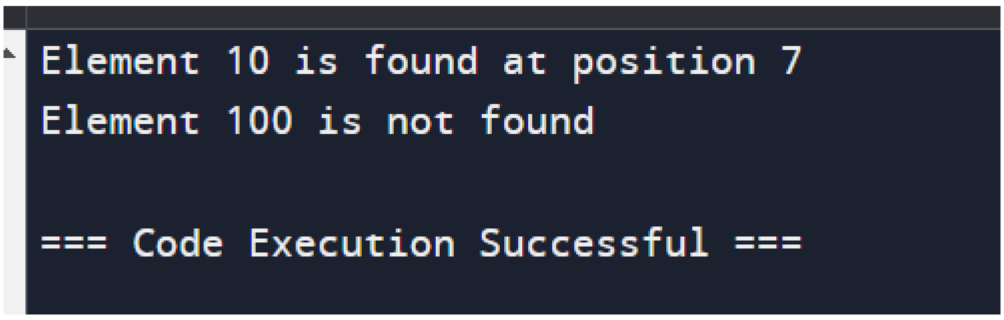
1. 133. Checks if a given number x exists in a sorted array arr using binary search. Analyze its time complexity using Big-O notation.
   Test Case:
   Example X={ 3,4,6,-9,10,8,9,30} KEY=10
   Output: Element 10 is found at position 5

   Example X={ 3,4,6,-9,10,8,9,30} KEY=100
   Output : Element 100 is not found

PROGRAM:-

```
def binary_search(arr, key):

    # Make sure the array is sorted

    arr.sort()

    left, right = 0, len(arr) - 1



    while left <= right:

        mid = left + (right - left) // 2



        # Check if the key is present at mid

        if arr[mid] == key:

            return mid



        # If key is greater, ignore the left half

        elif arr[mid] < key:

            left = mid + 1



        # If key is smaller, ignore the right half

        else:

            right = mid - 1



    # Element is not present in the array
    return -1


# Test Cases

# Test Case 1

arr1 = [3, 4, 6, -9, 10, 8, 9, 30]

key1 = 10

index1 = binary_search(arr1, key1)

if index1 != -1:

    print(f"Element {key1} is found at position {index1 + 1}")

else:

    print(f"Element {key1} is not found")


# Test Case 2

arr2 = [3, 4, 6, -9, 10, 8, 9, 30]

key2 = 100

index2 = binary_search(arr2, key2)

if index2 != -1:

    print(f"Element {key2} is found at position {index2 + 1}")

else:

    print(f"Element {key2} is not found")
```

OUTPUT:-



```
Element 10 is found at position 7
Element 100 is not found

=== Code Execution Successful ===
```

TIME COMPLEXITY:-O(logn)