# 3.8 BINARY SEARCH IMPLEMENTATION – STEPWISE TRACE & ANALYSIS

**Question:**

To implement Binary Search in Python, trace the mid-point calculations while searching for the element, and analyze the impact of unsorted input on correctness and performance.

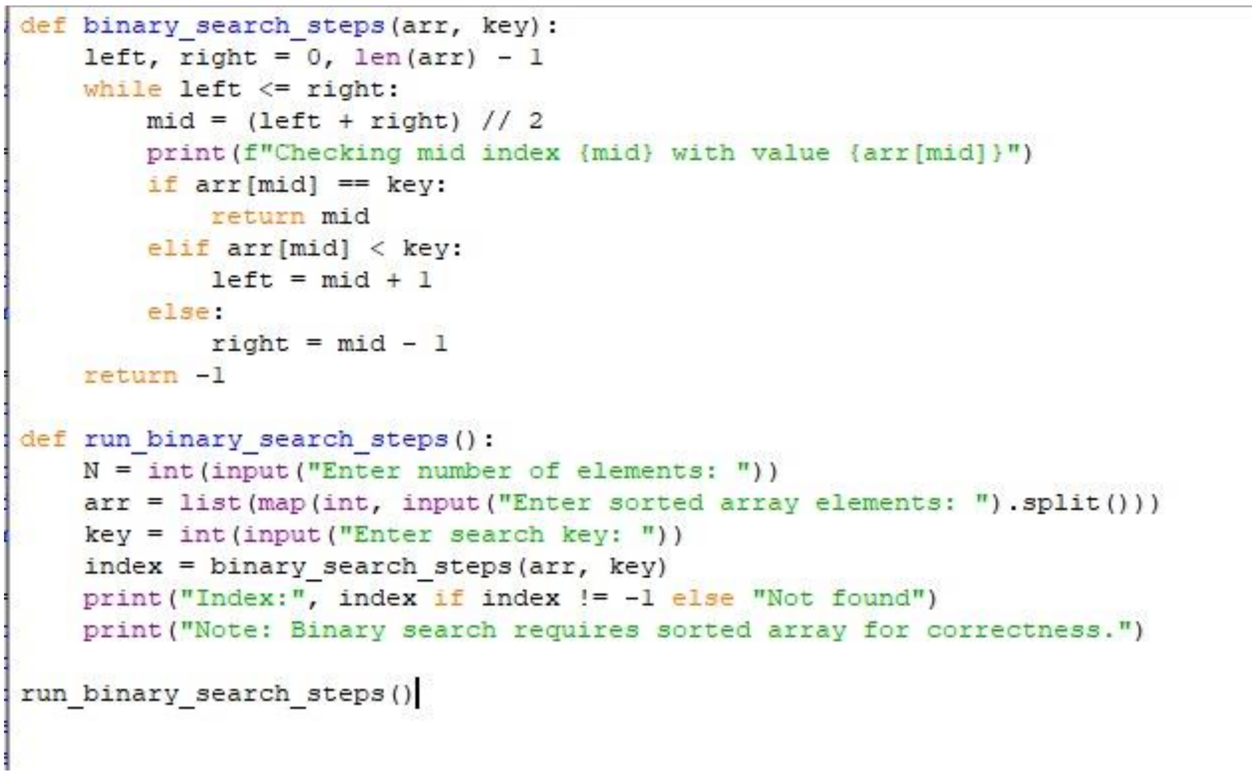
**AIM**

To return all strings from a given list that are substrings of another string in the same list.

**ALGORITHM**

1. Initialize low = 0, high = len(arr) - 1.
2. While low <= high:
    - Compute mid = (low + high) // 2.
    - Compare arr[mid] with target:
    - If equal → return index.
    - If target < arr[mid] → search left half.
    - If target > arr[mid] → search right half.
3. If not found → return -1.

**PROGRAM**



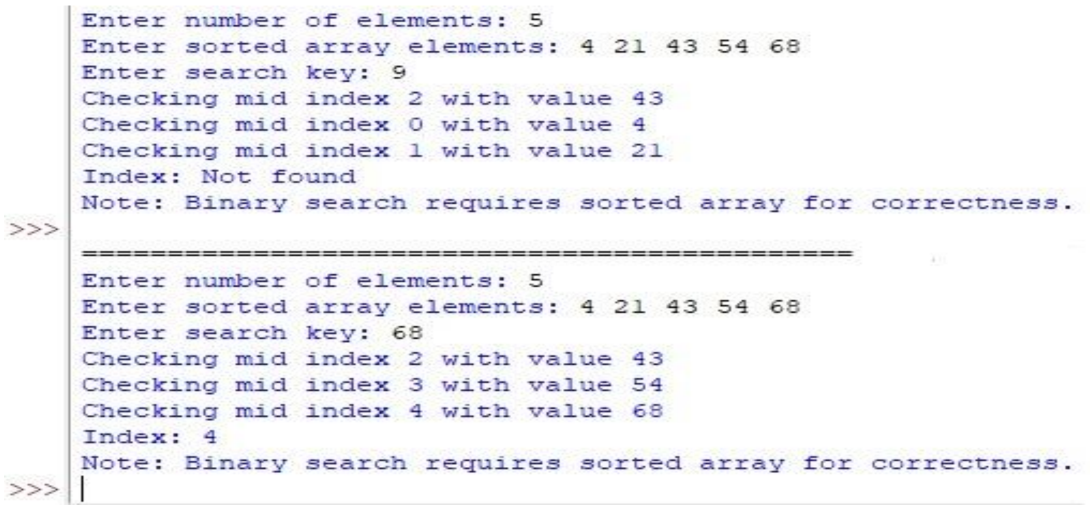
```python
def binary_search_steps(arr, key):
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        print(f"Checking mid index {mid} with value {arr[mid]}")
        if arr[mid] == key:
            return mid
        elif arr[mid] < key:
            left = mid + 1
        else:
            right = mid - 1
    return -1

def run_binary_search_steps():
    N = int(input("Enter number of elements: "))
    arr = list(map(int, input("Enter sorted array elements: ").split()))
    key = int(input("Enter search key: "))
    index = binary_search_steps(arr, key)
    print("Index:", index if index != -1 else "Not found")
    print("Note: Binary search requires sorted array for correctness.")

run_binary_search_steps()
```

Input:

N=5, [4,21,43,54,68]

key=9,68

Output:



```
Enter number of elements: 5
Enter sorted array elements: 4 21 43 54 68
Enter search key: 9
Checking mid index 2 with value 43
Checking mid index 0 with value 4
Checking mid index 1 with value 21
Index: Not found
Note: Binary search requires sorted array for correctness.
>>>
==============================================
Enter number of elements: 5
Enter sorted array elements: 4 21 43 54 68
Enter search key: 68
Checking mid index 2 with value 43
Checking mid index 3 with value 54
Checking mid index 4 with value 68
Index: 4
Note: Binary search requires sorted array for correctness.
>>>
```

**RESULT:**

Thus the program is successfully executed and the output is verified.

**PERFORMANCE ANALYSIS:**

· Time Complexity: O(log n)

· Space Complexity: O(1)