1. Program for Search Insert Position

```java
public class SearchInsertPosition {

    public static int searchInsert(int[] nums, int target) {
        int low = 0;
        int high = nums.length - 1;
        if (low > high) {
            return low; // Target not found, return position to insert
        }
        while (low <= high) {
            int mid = low + (high - low) / 2;

            if (nums[mid] == target) {
                return mid; // Target found at index mid
            } else if (nums[mid] < target) {
                low = mid + 1; // Adjust the lower bound
            } else {
                high = mid - 1; // Adjust the upper bound
            }
        }

        // Target not found, return the position where it should be inserted
        return low;
    }

    public static void main(String[] args) {
        int[] arr = {0,1,2,4,5,6};
        int target = 4;

        int position = searchInsert(arr, target);

        System.out.println("Position = "+ position);
    }
}
```

OUTPUT

Position = 3

2. Program for Substring with Concatenation of All Words

```
public class SubstringConcatenation {

    public static List<Integer> findSubstring(String s, String[] words) {
        List<Integer> result = new ArrayList<>();

        // Check for valid inputs
        if (s == null || s.length() == 0 || words == null || words.length == 0) {
            return result;
        }

        int wordLength = words[0].length();
        int wordsCount = words.length;
        int totalLength = wordLength * wordsCount;

        // Create a frequency map for words in the words array
        HashMap<String, Integer> wordMap = new HashMap<>();
        for (String word : words) {
            if (!wordMap.containsKey(word)) {
                wordMap.put(word, 1);
            } else {
                wordMap.put(word, wordMap.get(word) + 1);
            }
        }

        // Iterate through the string to find possible substrings
        for (int i = 0; i <= s.length() - totalLength; i++) {
            HashMap<String, Integer> showWords = new HashMap<>();
            int j = 0;

            // Check if concatenated words form valid substring
            while (j < wordsCount) {
                String word = s.substring(i + j * wordLength, i + (j + 1) *
wordLength);

                if (wordMap.containsKey(word)) {
                    showWords.put(word, showWords.getOrDefault(word, 0) + 1);

                    if (showWords.get(word) > wordMap.get(word)) {
                        break; // Break if word count exceeds allowed frequency
                    }
                } else {
                    break; // Break if word not found in wordMap
                }

                j++;
            }

            // If all words are found, add the starting index to the result
            if (j == wordsCount) {
                result.add(i);
            }
```

```
        }

        return result;
    }

    public static void main(String[] args) {
        String s = "barfoothefoobarmanbarfoodff";
        String[] words = {"foo", "bar"};

        List<Integer> result = findSubstring(s, words);

        System.out.println("Substrings Position = " + result);
    }
}
```

OUTPUT

Substrings Position = [0, 9, 18]