Daily Practice Problem Date 30/11/2023

Problem 1: Longest Word In The Dictionary

```
package ishwarchavan.com;
import java.util.*;

public class LongWordInDictionary {   //class created

    static ArrayList longestWords(String[] dictionary) {  // Function to find and
return the longest words in the given dictionary
        ArrayList list = new ArrayList();
        int longest_length = 0;

        for (String str : dictionary) {   // Iterate through each word in the
dictionary
            int length = str.length();

            if (length > longest_length) {   // Check if the current word is longer
than the previously found longest word(s)
                longest_length = length;
                list.clear(); // Clear the list as a new longest word is found
            }

            if (length == longest_length) {   // If the current word has the same
length as the longest word(s), add it to the list
                list.add(str);
            }
        }
        return list; // Return the list of longest words
    }

    public static void main(String[] args) {     //main program started
        String[] dict = {"i", "ish", "ishw", "ishwa", "ishwar"};
        System.out.println("Longest word in the given dictionary dictionary: " +
longestWords(dict));
    }
}
```

Problem 2: Increasing Order Search Tree

```
package ishwarchavan.com;
import java.util.*;

public class IncreasingOrderSearchTree {    //class created

static class node{   //Node of the binary tree
int data;
node left;
node right;

node(int data){    //node created
      this.data = data;
      left = null;
      right = null;
 }
};


static void print(node parent){   //Function to print flattened binary tree
node curr = parent;
while (curr != null){   //condition checking
      System.out.print(curr.data + " ");
      curr = curr.right;
```

```
 }
}

static node prev;

static void Inorder(node curr){  //Function to perform in-order traversal Base case
if (curr == null)  //if true then return
      return;
Inorder(curr.left);   //function calling
prev.left = null;
prev.right = curr;
prev = curr;
Inorder(curr.right);   //function calling
 }


static node flatten(node parent){  //Function to flatten binary tree using level order
traversal

node dummy = new node(-1);  //Dummy node
prev = dummy;  //Pointer to previous element
Inorder(parent);  //Calling in-order traversal

prev.left = null;
prev.right = null;
node ret = dummy.right;
return ret;
 }

public static void main(String[] args){    //main program started
node root = new node(4);
root.left = new node(2);
root.right = new node(6);
root.left.left = new node(1);
root.left.right = new node(3);
root.right.left = new node(5);
root.right.right = new node(7);

print(flatten(root));  //Calling required function
 }
}
```