Problem 1: Match Found

```
public class MatchWords {
    public static String matchFound(String input1, String input2) {
        String[] words = input2.split(":");
        String pattern = input1.replace('_', '.');
        StringBuilder output = new StringBuilder();
        for (String word : words) {
            if (word.matches(pattern)) {
                if (output.length() > 0) {
                    output.append(":");
                }
                output.append(word.toUpperCase());
            }
        }
        return output.toString();
    }
}
```

Problem 2: Encode Three Strings

```
public class EncodeThreeStrings {
    public static String[] encodeStrings(String input1, String input2, String input3) {
        String[] parts1 = splitString(input1);
        String[] parts2 = splitString(input2);
        String[] parts3 = splitString(input3);

        String output1 = parts1[0] + parts2[1] + parts3[2];
        String output2 = parts1[1] + parts2[2] + parts3[0];
        String output3 = parts1[2] + parts2[0] + parts3[1];

        output3 = toggleCase(output3);
        return new String[]{output1, output2, output3};
    }

    private static String[] splitString(String input) {
        int len = input.length();
        int part = len / 3, rem = len % 3;
        int f = part + (rem == 2 ? 1 : 0);
        int m = part + (rem == 1 ? 1 : 0);
        int e = len - f - m;
              return  new  String[]{input.substring(0,  f),  input.substring(f,  f  +  m),
input.substring(f + m)};
    }

    private static String toggleCase(String str) {
        StringBuilder sb = new StringBuilder();
        for (char c : str.toCharArray()) {
                        sb.append(Character.isUpperCase(c)  ?  Character.toLowerCase(c)  :
Character.toUpperCase(c));
        }
        return sb.toString();
    }
}
```

Problem 3: Append Alphabet

```
public class AppendAlphabet {
    public static String transformString(String input) {
        StringBuilder output = new StringBuilder();
        input = input.toLowerCase();
        for (int i = 0; i < input.length(); i++) {
            char ch = input.charAt(i);
            output.append(ch);
                        if  (Character.isLetter(ch)  &&  i  +  1  <  input.length()  &&
Character.isLetter(input.charAt(i + 1))) {
                int val1 = ch - 'a' + 1;
                int val2 = input.charAt(i + 1) - 'a' + 1;
                int sum = (val1 + val2) % 26;
                output.append(sum == 0 ? '0' : (char)('a' + sum - 1));
            }
        }
        return output.toString();
    }
}
```

Problem 4: Find Added Letter

```
public class FindTheDifference {
    public static char findTheDifference(String s, String t) {
        int result = 0;
        for (char c : s.toCharArray()) result ^= c;
        for (char c : t.toCharArray()) result ^= c;
        return (char) result;
    }
}
```

Problem 5: Next Greater Element

```
import java.util.*;
public class NextGreaterElement {
    public int[] nextGreaterElement(int[] nums1, int[] nums2) {
        Map<Integer, Integer> map = new HashMap<>();
        Stack<Integer> stack = new Stack<>();
        for (int num : nums2) {
            while (!stack.isEmpty() && stack.peek() < num) {
                map.put(stack.pop(), num);
            }
            stack.push(num);
        }
        int[] res = new int[nums1.length];
        for (int i = 0; i < nums1.length; i++) {
            res[i] = map.getOrDefault(nums1[i], -1);
        }
        return res;
    }
}
```

Problem 6: Balanced Parentheses

```
import java.util.*;
```

```
public class BalancedParentheses {
    public static boolean isBalanced(String s) {
        Stack<Character> stack = new Stack<>();
        for (char c : s.toCharArray()) {
            if (c == '(' || c == '{' || c == '[') {
                stack.push(c);
            } else {
                if (stack.isEmpty()) return false;
                char open = stack.pop();
                 if ((c == ')' && open != '(') || (c == '}' && open != '{') || (c == ']'
&& open != '[')) return false;
            }
        }
        return stack.isEmpty();
    }
}
```

Problem 7: Player Comparator

```
import java.util.*;
class Player {
    String name;
    int score;
    Player(String name, int score) {
        this.name = name;
        this.score = score;
    }
}

class Checker implements Comparator<Player> {
    public int compare(Player a, Player b) {
        if (a.score != b.score) return b.score - a.score;
        return a.name.compareTo(b.name);
    }
}
```

Problem 8: BigDecimal Sort

```
import java.math.BigDecimal;
import java.util.*;

public class BigDecimalSort {
    public static void sortBigDecimals(String[] s) {
        Arrays.sort(s, (a, b) -> new BigDecimal(b).compareTo(new BigDecimal(a)));
    }
}
```

Problem 9: Wildcard Matching

```
public class WildcardMatching {
    public boolean isMatch(String s, String p) {
        boolean[][] dp = new boolean[s.length() + 1][p.length() + 1];
        dp[0][0] = true;
        for (int j = 1; j <= p.length(); j++) {
            if (p.charAt(j - 1) == '*') dp[0][j] = dp[0][j - 1];
        }
```

```
        for (int i = 1; i <= s.length(); i++) {
            for (int j = 1; j <= p.length(); j++) {
                if (p.charAt(j - 1) == '*') {
                    dp[i][j] = dp[i][j - 1] || dp[i - 1][j];
                } else if (p.charAt(j - 1) == '?' || s.charAt(i - 1) == p.charAt(j - 1))
{
                    dp[i][j] = dp[i - 1][j - 1];
                }
            }
        }
        return dp[s.length()][p.length()];
    }
}
```

Problem 10: Find Range

```
public class FindRange {
    public int[] searchRange(int[] nums, int target) {
        int left = findBound(nums, target, true);
        int right = findBound(nums, target, false);
        return new int[]{left, right};
    }

    private int findBound(int[] nums, int target, boolean isFirst) {
        int low = 0, high = nums.length - 1, result = -1;
        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (nums[mid] == target) {
                result = mid;
                if (isFirst) high = mid - 1;
                else low = mid + 1;
            } else if (nums[mid] < target) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }
        return result;
    }
}
```