Assignment (Fast Learner)

Student Name: Aditya Prabhakar **UID:** 22BCS12889
Branch: CSE **Section/Group:** IOT-618/A
Semester: 6th **Date of Submission:** 10/04/25
Subject Name: PBLJ **Subject Code:** 22CSH-359

Problem-1

Consider a function public String matchFound(String input 1, String input 2), where

- ● input1 will contain only a single word with only 1 character replaces by an underscore '_'
- ● input2 will contain a series of words separated by colons and no space character in between
- ● input2 will not contain any other special character other than underscore and alphabetic characters.

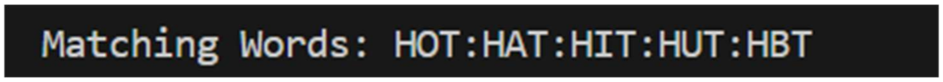
The methods should return output in a String type variable "output1" which contains all the words from input2 separated by colon which matches with input 1. All words in output1 should be in uppercase.

Code:

```
public class Solution {

  public static String matchFound(String input1, String input2) {
    String[] words = input2.split(":");
    StringBuilder output1 = new StringBuilder();

    for (String word : words) {
      if (word.length() == input1.length()) {
        boolean match = true;
        for (int i = 0; i < input1.length(); i++) {
          if (input1.charAt(i) != '_' && input1.charAt(i) != word.charAt(i)) {
            match = false;
            break;
```

```
                }
            }
            if (match) {
                if (output1.length() > 0) output1.append(":");
                output1.append(word.toUpperCase());
            }
        }
    }

    return output1.toString();
}

public static void main(String[] args) {
    String input1 = "h_t";
    String input2 = "hot:hat:hit:hut:hbt";

    String result = matchFound(input1, input2);
    System.out.println("Matching Words: " + result);
}
}
```

Output:



```
Matching Words: HOT:HAT:HIT:HUT:HBT
```

Problem-2

Given a String (In Uppercase alphabets or Lowercase alphabets), new alphabets is to be appended with following rule:

(i) If the alphabet is present in the input string, use the numeric value of that alphabet. E.g. a or A numeric value is 1 and so on. New alphabet to be appended between 2 alphabets:

(a) If (sum of numeric value of 2 alphabets) %26 is 0, then append 0. E.g. string is ay. Numeric value of a is 1, y is 25. Sum is 26. Remainder is 0, the new string will be a0y.

(b) Otherwise (sum of numeric value of 2 alphabets) %26 numeric value alphabet is to be appended. E.g. ac is string. Numeric value of a is 1, c is 3, sum is 4. Remainder with 26 is 4. Alphabet to be appended is d. output will be adc.

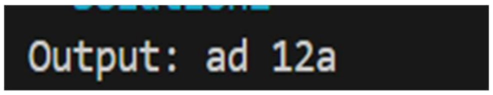
(ii) If a digit is present, it will be the same in the output string. E.g. string is 12, output string is 12.
(iii) If only a single alphabet is present, it will be the same in the output string. E.g. input string is 1a, output will be 1a.
(iv) If space is present, it will be the same in the output string. E.g. string is ac 12a, output will be adc 12a. Constraint: Whether string alphabets are In Uppercase or Lowercase, appended alphabets must be in lower case. Output string must also be in lowercase.

Code:

```
public class Solution2 {

  public static int getCharValue(char ch) {
    if (Character.isLetter(ch)) {
      return Character.toLowerCase(ch) - 'a' + 1;
    }
    return 0;
  }

  public static String processString(String input) {
    StringBuilder result = new StringBuilder();
    input = input.toLowerCase();

    int i = 0;
    while (i < input.length()) {
      char current = input.charAt(i);

      if (!Character.isLetter(current) || i == input.length() - 1) {
        result.append(current);
        i++;
        continue;
      }

      char next = input.charAt(i + 1);

      if (Character.isLetter(next)) {
        int sum = getCharValue(current) + getCharValue(next);
        if (sum % 26 == 0) {
          result.append(current).append("0");
        } else {
```

```
                    char toInsert = (char) ('a' + (sum % 26) - 1);
                    result.append(current).append(toInsert);
                }
                i++;
            } else {
                result.append(current);
            }

            i++;
        }

        return result.toString();
    }

    public static void main(String[] args) {
        String input = "ac 12a";
        String output = processString(input);
        System.out.println("Output: " + output);  // adc 12a
    }
}
```

Output:



```
Output: ad 12a
```

Problem-3

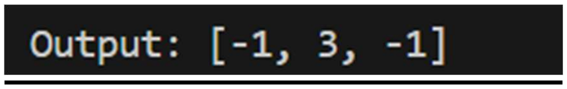
The next greater element of some element x in an array is the first greater element that is to the right of x in the same array.
You are given two distinct 0-indexed integer arrays nums1 and nums2, where nums1 is a subset of nums2.
For each 0 <= i < nums1.length, find the index j such that nums1[i] == nums2[j] and determine the next greater element of nums2[j] in nums2. If there is no next greater element, then the answer for this query is -1.
Return an array ans of length nums1.length such that ans[i] is the next greater element as described above.

Hint:
Input: nums1 = [4,1,2], nums2 = [1,3,4,2]
Output: [-1,3,-1]
Explanation: The next greater element for each value of nums1 is as follows: - 4 is underlined in nums2 = [1,3,4,2]. There is no next greater element, so the answer is -1. - 1 is underlined in nums2 = [1,3,4,2]. The next greater element is 3. - 2 is underlined in nums2 = [1,3,4,2]. There is no next greater element, so the answer is -1.

Code:

```
import java.util.*;

public class Solution3 {

    public static int[] nextGreaterElement(int[] nums1, int[] nums2) {
        Map<Integer, Integer> nextGreaterMap = new HashMap<>();
        Stack<Integer> stack = new Stack<>();

        for (int num : nums2) {
            while (!stack.isEmpty() && num > stack.peek()) {
                nextGreaterMap.put(stack.pop(), num);
            }
            stack.push(num);
        }

        while (!stack.isEmpty()) {
            nextGreaterMap.put(stack.pop(), -1);
        }

        int[] result = new int[nums1.length];
        for (int i = 0; i < nums1.length; i++) {
            result[i] = nextGreaterMap.get(nums1[i]);
        }

        return result;
    }

    public static void main(String[] args) {
        int[] nums1 = {4, 1, 2};
        int[] nums2 = {1, 3, 4, 2};
        int[] output = nextGreaterElement(nums1, nums2);

        System.out.println("Output: " + Arrays.toString(output));  // Output: [-1, 3, -1]
    }
}
```

Output:



```
Output: [-1, 3, -1]
```

Problem-4

Comparators are used to compare two objects. In this challenge, you'll create a comparator and use it to sort an array.
The Player class has fields: a String and a integer.
Given an array of Player objects, write a comparator that sorts them in order of decreasing score; if or more players have the same score, sort those players alphabetically by name.
To do this, you must create a Checker class that implements the Comparator interface, then write an int compare(Player a, Player b) method implementing the Comparator. compare(T o1, T o2) method.

Input Format
The first line contains an integer, denoting the number of players. Each of the subsequent lines contains a player's and , respectively.

Constraints

- players can have the same name.
- Player names consist of lowercase English letters.

Sample Input
5
amy 100
david 100
heraldo 50
aakansha 75
aleksa 150

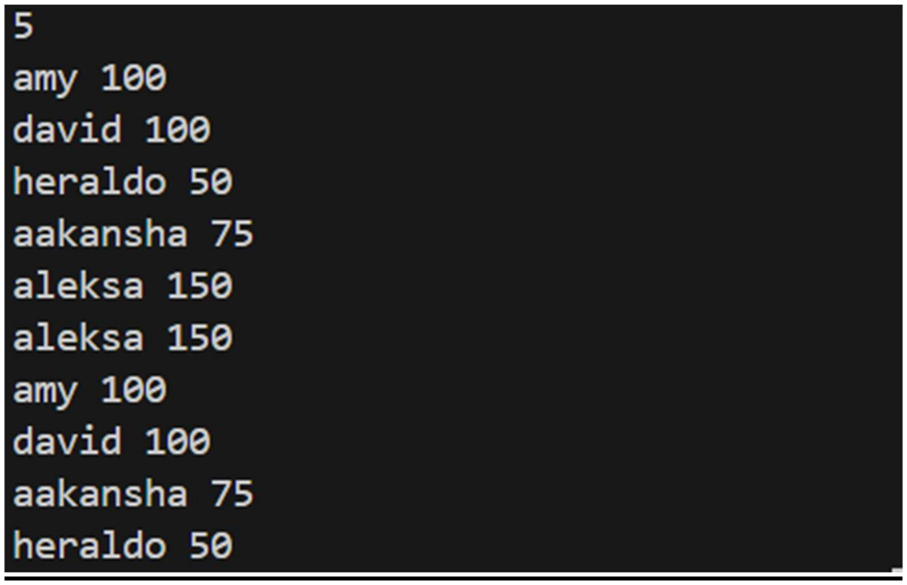
Sample Output

```
aleksa 150
amy 100
david 100
aakansha 75
heraldo 50
```

Code:

```
import java.util.*;

// Player class
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;
        } else {
            return a.name.compareTo(b.name);
        }
    }
}

public class Solution4 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        Player[] players = new Player[n];
        for (int i = 0; i < n; i++) {
            String name = sc.next();
            int score = sc.nextInt();
            players[i] = new Player(name, score);
        }

        Arrays.sort(players, new Checker());

        for (Player p : players) {
            System.out.println(p.name + " " + p.score);
        }

        sc.close();
    }
}
```

Output:



```
5
amy 100
david 100
heraldo 50
aakansha 75
aleksa 150
aleksa 150
amy 100
david 100
aakansha 75
heraldo 50
```

Problem-5

Given an input string (s) and a pattern (p), implement wildcard pattern matching with support for '?' and '*'
where:

- '?' Matches any single character.
- '*' Matches any sequence of characters (including the empty sequence).

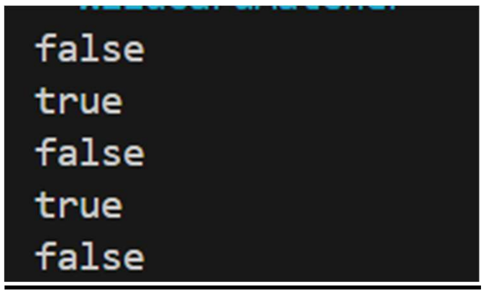
The matching should cover the entire input string (not partial).
Example 1:
Input: s = "aa", p = "a"
Output: false
Explanation: "a" does not match the entire string "aa".
Constraints:

- 0 <= s.length, p.length <= 2000
- s contains only lowercase English letters.
- p contains only lowercase English letters, '?' or '*'.

Code:

```
public class WildcardMatcher {

    public static boolean isMatch(String s, String p) {
        int m = s.length(), n = p.length();

        // dp[i][j] = true if s[0..i-1] matches p[0..j-1]
        boolean[][] dp = new boolean[m + 1][n + 1];
        dp[0][0] = true; // empty pattern matches empty string

        for (int j = 1; j <= n; j++) {
            if (p.charAt(j - 1) == '*')
                dp[0][j] = dp[0][j - 1];
        }
        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                char sc = s.charAt(i - 1);
                char pc = p.charAt(j - 1);

                if (pc == '*') {
                    dp[i][j] = dp[i][j - 1] || dp[i - 1][j];
                } else if (pc == '?' || sc == pc) {
                    dp[i][j] = dp[i - 1][j - 1];
                }
            }
        }

        return dp[m][n];
    }

    public static void main(String[] args) {
        String s = "aa";
        String p = "a";

        System.out.println(isMatch(s, p)); // Output: false

        System.out.println(isMatch("aa", "*"));      // Output: true
        System.out.println(isMatch("cb", "?a"));     // Output: false
        System.out.println(isMatch("adceb", "*a*b")); // Output: true
        System.out.println(isMatch("acdcb", "a*c?b")); // Output: false
    }
}
```

Output:



```
false
true
false
true
false
```