

Department: BE-CSE/IT 3rd Year

Name: Namandeep Jain

Uid 22BCS14946

Sec/Group: 22BCS-IOT-618-A

Subject: Project Based Learning in Java

Subject Code: 22CSH-359

Semester: 6th

Batch: 2022

Lab Based Complex Coding Problems

Problem 1.

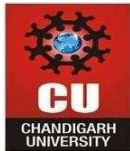
Consider a function **public String matchFound(String input 1, String input 2)**, where

- **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.

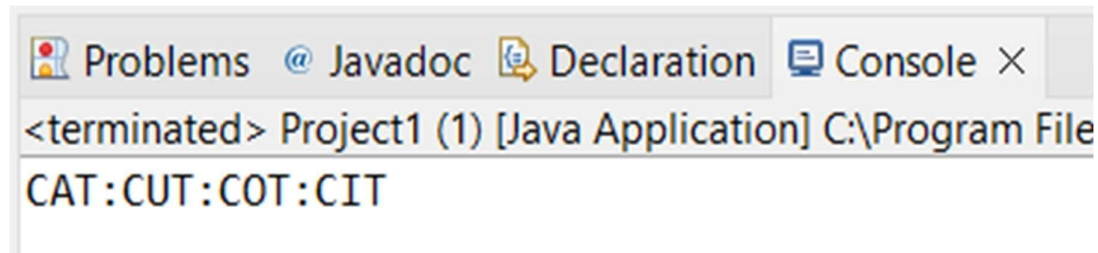
The methods should return output in a String type variable “**output1**” which contains all the words from input2 separated by colon which matches with input 1. All words in output1 should be in uppercase.

Code:

```
public class Project1 {
    public static void main(String[] args)
    {
        String input1 = "c_t";
        String input2 = "cat:bat:cut:cot:cit:mat:rat";
        String output1 = matchFound(input1, input2);
        System.out.println(output1);
    }
    public static String matchFound(String input1, String input2) {
        String[] words = input2.split(":");
        StringBuilder result = new StringBuilder();
        int underscoreIndex = input1.indexOf('_');
        for (String word : words) {
            if (word.length() == input1.length())
            {
                boolean match = true;
                for (int i = 0; i < word.length(); i++) {
                    if (i != underscoreIndex && input1.charAt(i) != word.charAt(i))
                    {
                        match = false;
                        break;
                    }
                }
                if (match) {
                    if (result.length() > 0)
                    {
                        result.append(":");
                    }
                    result.append(word.toUpperCase());
                }
            }
        }
        return result.toString();
    }
}
```



Output:



Problem 3:

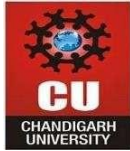
Given a String (In Uppercase alphabets or Lowercase alphabets), new alphabets is to be appended with following rule:

- (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.

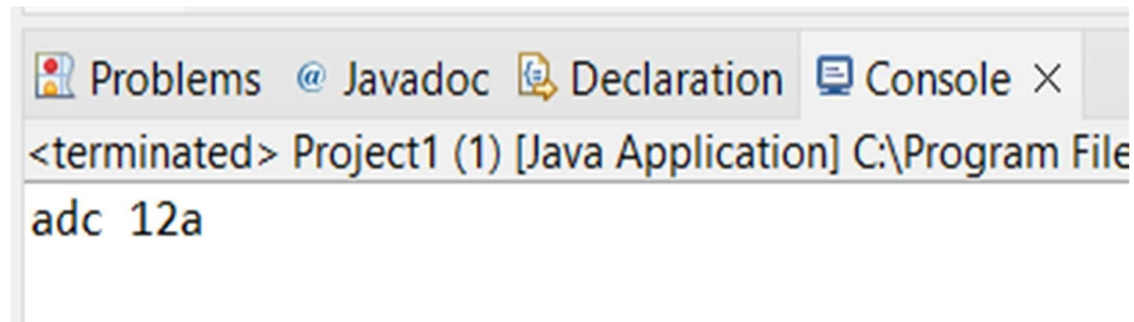
Code:

```
public class Project1 {
    public static void main(String[] args)
    {
        String input = "ac 12a";
        String output = buildNewString(input);
        System.out.println(output);
    }
    public static String buildNewString(String input)
    {
        StringBuilder result = new StringBuilder();
        int n = input.length();
        for (int i = 0; i < n; i++) {
            char current = input.charAt(i);
            result.append(Character.toLowerCase(current));
            if (i < n - 1) {
                char next = input.charAt(i + 1);
                if (Character.isLetter(current) && Character.isLetter(next)) {
```



```
int val1 = Character.toLowerCase(current) - 'a' + 1;
int val2 = Character.toLowerCase(next) - 'a' + 1;
int sum = (val1 + val2) % 26;
if (sum == 0)
{ result.append("0");
} else {
    result.append((char) ('a' + sum - 1));
}
}
}
}
return result.toString();
}
}
```

Output:



Problem 5:

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 $0 \leq i < nums1.length$, 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:

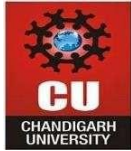
Input: $nums1 = [4,1,2]$, $nums2 = [1,3,4,2]$ Output: $[-1,3,-1]$

Explanation: The next greater element for each value of $nums1$ is as follows:

- 4 is underlined in $nums2 = [1,3,\underline{4},2]$. There is no next greater element, so the answer is -1.
- 1 is underlined in $nums2 = [\underline{1},3,4,2]$. The next greater element is 3.
- 2 is underlined in $nums2 = [1,3,4,\underline{2}]$. There is no next greater element, so the answer is -1.

Code:

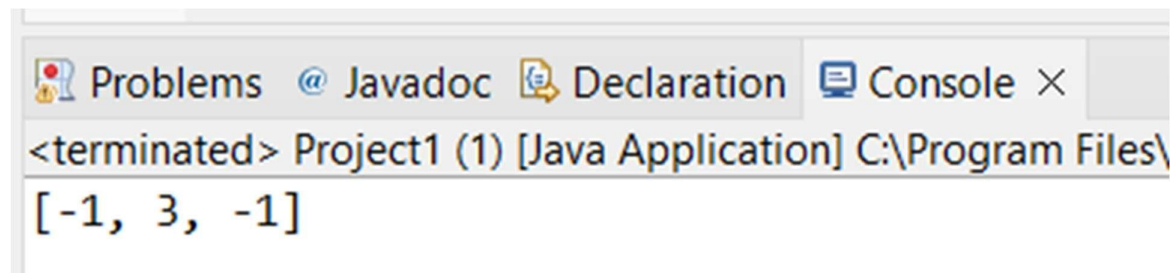
```
import java.util.*;
public class Project1 {
    public static void main(String[] args) { int[]
        nums1 = {4, 1, 2};
```



```
int[] nums2 = {1, 3, 4, 2};
int[] result = nextGreaterElement(nums1, nums2);
System.out.println(Arrays.toString(result));
}

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;
}
}
```

Output:



Problem 7:

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.

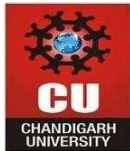
To do this, you must create a Checker class that implements the Comparator interface, then write an int compare(Player a, Player b) method implementing the Comparator.compare(T o1, T o2) method.

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**

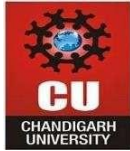
5
amy 100
david 100
heraldo 50
aakansha 75
aleksa 150

Sample Output

aleksa 150
amy 100
david 100
aakansha 75
heraldo 50

Code:

```
import java.util.*;
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 Project1 {
    public static void main(String[] args)
    { Scanner sc = new Scanner(System.in);
      System.out.print("Enter number of players: ");
      int n = sc.nextInt();
      sc.nextLine();
      Player[] players = new Player[n];
      System.out.println("Enter player name and score (e.g., amy 100):");
      for (int i = 0; i < n; i++) {
          String line = sc.nextLine();
          String[] parts = line.split(" ");
          String name = parts[0];
          int score = Integer.parseInt(parts[1]);
          players[i] = new Player(name, score);
      }
      Arrays.sort(players, new Checker());
      System.out.println("\nSorted Players:");
      for (Player p : players) {
          System.out.println(p.name + " " + p.score);
      }
      sc.close();
    }
}
```

**Output:**

```
Problems @ Javadoc Declaration Console ×
<terminated> Project1 (1) [Java Application] C:\Program Files
Enter number of players: 5
Enter player name and score (e.g., amy 100):
amy 100
david 100
heraldo 50
aakansha 75
aleksa 150

Sorted Players:
aleksa 150
amy 100
david 100
aakansha 75
heraldo 50
```

Problem 9:

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

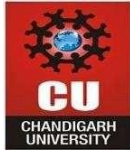
Explanation: "a" does not match the entire string "aa".

Constraints:

- $0 \leq s.length, p.length \leq 2000$
- s contains only lowercase English letters.
- p contains only lowercase English letters, '?' or '*'.

Code:

```
public class Project1 {
    public static void main(String[] args)
    {
        String s = "aa";
        String p = "a";
        boolean result = isMatch(s, p);
        System.out.println(result);
    }
    public static boolean isMatch(String s, String p)
    {
        int m = s.length();
        int n = p.length();
        boolean[][] dp = new boolean[m + 1][n + 1];
```



```
dp[0][0] = true;
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 pc = p.charAt(j - 1);
    char sc = s.charAt(i - 1);
    if (pc == '*') {
        dp[i][j] = dp[i][j - 1] || dp[i - 1][j];
    } else if (pc == '?' || pc == sc)
        { dp[i][j] = dp[i - 1][j - 1];
        }
    }
}
return dp[m][n];
}
```

Output:

