1. **DS & Algo Problems**

NOTE: Please make sure if using java 8 streams in the coder pad editor, the package needs to be imported manually.

* Longest Word:

Given a string of letters and a dictionary, the function longest word should

find the longest word or words in the a dictionary that can be made from the letters

Input: letters = "oet", dictionary = {"to","toe","toes"}

Output: {"toe"}

Only lowercase letters will occur in the dictionary and the letters

The length of letters will be between 1 and 10 characters

The solution should work well for a dictionary of over 100,000 words

* Decimal Conversion  
  Implement the method provided numerator and denominator will return a string representing fraction's decimal form. Some fractions in decimal form have cyclic decimal points.

Implement the method that provided numerator and denominator will return a string representing fraction's decimal form.

                Some fractions in decimal form have cyclic decimal points.

public static String vulgarToDecimal(Long numerator, Long denominator) {

}

|  |
| --- |
| 1. Longest word:  Problem Statement-  Given a string of letters and a dictionary, the function longestWord should  find the longest word or words in the dictionary that can be made from the letters  Input: letters = "oet", dictionary = {"to","toe","toes"}  Output: {"toe"}  Only lowercase letters will occur in the dictionary and the letters  The length of letters will be between 1 and 10 characters                   The solution should work well for a dictionary of over 100,000 words  Signature- Set<String> longestWord(String letters, Dictionary dictionary) {                      }  Test Cases-  Dictionary dict = new Dictionary(new String[]{"to", "toe", "toes", "doe", "dog", "god", "dogs", "book", "banana"});     Input- toe  2. Search Tree  Problem Statement:  Implement the “put" and “contains” methods.                  Fix the "“inOrderTraversal” method.                      public class Problem {      static class BST {          private Node root;          public BST() {              this.root = new Node();          }          public void put(int value) {              // TODO: implement me          }          public boolean contains(int value) {              // TODO: implement me              return false;          }          public List<Integer> inOrderTraversal() {              final ArrayList<Integer> acc = new ArrayList<>();              inOrderTraversal(root, acc);              return acc;          }          private void inOrderTraversal(Node node, List<Integer> acc) {              if (node == null) {                  return;              }              inOrderTraversal(node.left, acc);              inOrderTraversal(node.right, acc);              acc.add(node.val);          }          private static class Node {              Integer val;              Node left;              Node right;          }      }      public static void testBST() {          final BST searchTree = new BST();          searchTree.put(3);          searchTree.put(1);          searchTree.put(2);          searchTree.put(5);          assertFalse(searchTree.contains(0));          assertTrue(searchTree.contains(1));          assertTrue(searchTree.contains(5));          assertFalse(searchTree.contains(6));          assertEquals(Arrays.asList(1, 2, 3, 5), searchTree.inOrderTraversal());      }      private static void assertFalse(boolean rez) {          if (rez) {              throw new RuntimeException("Test failed");          } else {              System.out.println("Test passed");          }      }      private static void assertTrue(boolean rez) {          if (!rez) {              throw new RuntimeException("Test failed");          } else {              System.out.println("Test passed");          }      }      private static void assertEquals(List<Integer> expected, List<Integer> result) {          if (!expected.equals(result)) {              System.out.println(String.format("Test failed \"%s\" not equeals to \"%s\"", expected, result));          } else {              System.out.println("Test passed");          }      }      // TODO: write some more tests      public static void main(String[] args) {          testBST();      }  } |

|  |
| --- |
| /\*\*    \* char findFirst(String input)    \* Finds the first character that does not repeat anywhere in the input string    \* If all characters are repeated, return 0    \* Given "apple", the answer is "a"    \* Given "racecars", the answer is "e"    \* Given "ababdc", the answer is "d"    \*\*/ |

|  |
| --- |
| 1. Merge and sort array of linkedlists 2. Fraction to decimal conversion with recursing decimals |

|  |
| --- |
| Round 1- I Noticed a change in pattern(More Focus on Java and Springboot than DSA) First question -  Length of longest substring without repeating characters String s = "abcabcd" --> 4, 3, 6 () Maxlength,startPosition , EndPosition  Only explain the algo don't write the code.  String s1 = "abc"; String s2 = "abc"; String s3 = new String("abc");  s1 == s2 --> true s1 == s3 --> false s1.equals(s3) --> true s1.equals(s2) --> true  What is Object Class; Difference between Set clear vs Set reinitialize memory-wise and Time complexity wise Design question - Make a Small app using Rest API in Spring boot including COntroller , Bussiness Class,CrudRepository , Pojo ,Apicalls , Test Cases using Junit , Validation of Data in Spring boot (How will you do it ?)  class Person {  }\*/ |

|  |
| --- |
| 1. Find First Non repeated Char in a String Ex- "aapple" -> ans 'l' Follow up question : Find 3rd Non repeated char in String Ex- "aappletty" -> y  2. Find Optimal Path in Grid , where you can move only Up and right , Start from last row and first column and reach first row and last col and  matrix[m-1][0] -> matrix[0][n-1] and collect coins for maximum sum.  3. SnowPack - Given an array of non-negative integers representing the elevations from the vertical cross section of a range of hills, determine how many units of snow could be captured between the hills i/p - {0, 1, 3, 0, 1, 2, 0, 4, 2, 0, 3, 0}  o/p - 13 |

|  |
| --- |
| 1. **Search Tree** - Implement the “put" and “contains” methods and Fix the "“inOrderTraversal” method. 2. **Longest Word** - Given a string of letters and a dictionary, the function longestWord should find the longest word or words in the dictionary that can be made from the letters. |

## Count Length Of Cycle

|  |
| --- |
| Problem Statement:         You are given an integer array of size N.       Every element of the array is greater than or equal to 0.       Starting from arr[startIndex], follow each element to the index it points to.       Continue to do this until you find a cycle.       Return the length of the cycle. If no cycle is found return -1         Examples:      Signature:  Int countLengthofcycle(arr, startIndex){  }  Test Cases:  Input :  arr: [1,0]  startIndex: 0  Output :2  Input :  arr: [1,2,0]  startIndex: 0  Output :2 |

## Snowpack

|  |
| --- |
| Problem Statement:  Given an array of non-negative integers representing the elevations    from the vertical cross section of a range of hills, determine how    many units of snow could be captured between the hills.    See the example array and elevation map below.                                  \_\_\_              \_\_\_                |   |        \_\_\_             |   |        \_\_\_    |   |\_\_\_    |   |  \*        \_\_\_|   |    \_\_\_|   |   |   |   |   |   |      \_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_     { 0,  1,  3,  0,  1,  2,  0,  4,  2,  0,  3,  0 }                                  \_\_\_              \_\_\_                |   |        \_\_\_             |   | \*   \*  \_\*\_  \* |   |\_\*\_  \* |   |  \*        \_\_\_|   | \*  \_\*\_|   | \* |   |   | \* |   |      \_\_\_|\_\_\_|\_\_\_|\_\*\_|\_\_\_|\_\_\_|\_\*\_|\_\_\_|\_\_\_|\_\*\_|\_\_\_|\_\_\_     { 0,  1,  3,  0,  1,  2,  0,  4,  2,  0,  3,  0 }    Solution: In this example 13 units of snow (\*) could be captured.  Signature: public static Integer computeSnowpack(Integer[] arr) {  }  Test Cases:  Input : {0, 1, 3, 0, 1, 2, 0, 4, 2, 0, 3, 0}  Output: 13  Input : {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1}  Output:10 |

1. **Technical Questions (2nd Round)**

|  |
| --- |
| **Two questions on code snippets : asked what is wrong with the code and how to fix, what will be the output of those snippets ?**  **Questions related to my experience and my previous work.**  **Question on joins, self-join, inner join right joins, when and why we use inner join.**  **Question on restful services, get mapping and post mapping.**  **Question on deep copy and how to do that.**  **In what scenarios we use get and post ?**  **In what scenarios I use post mapping to get the data ?**  **How rest is different from other types of api’s ?**  **Questions on how rest is statelessness, and scenario-based question on that.**  Q1. Reverse Strinf without effecting special characters.  Q2. Find if a linked list is circular or not.  Q3. Class level design (LLD) for Marathon.  Interviewer: Hrishikesh  All the questions were asked from resume.  1. Rest principles  2. Spring boot vs Spring  3. Views / stored procedures / indexes  4. Why Rest is better than Soap?  5. In order to optimise the query fetching millions record what will be your approach. |

Question 1.  
input: "/gH?yZx"

output: "/xZ?yHg"

Question 2.  
26.2 miles race, multiple runner there  
at 10 miles make leaderboard of the top 10 runners

Question 3.  
reverse String

Question 4.  
check a string if palindrome or not

Questions 5.  
REST API, Dependency injection, IOC, REST is synchronous or asynchronous? If thousands of request coming to REST controller for same api, how will handle?