Score: 2.500 (=100.0%)
Id: 42300
U ovom zadatku se koristi zapis decimalnih brojeva na "distribuirani" način. Njihove znamenke su zapisane u listi sa oznakama pozicije. Detaljnije:

- Lista se sastoji od **podlista** koje sadrže po dva elementa (**a**,**b**).
- pritom **a** označava redni broj decimalnog mjesta (pozicije) na sljedeći način: 0.1234567. 0. je zadnje mjesto ispred decimalne točke, a potom ostali redni brojevi označavaju pozicije na desno od decimalne točke. Pozicije lijevo od 0. se ignoriraju.
- pritom **b** je znamenka na mjestu označenom sa pripadnim **a**.

Primjer: broj 3.141 zapisan na distribuirani način jest: ((0,3), (1,1), (2,4), (3,1))

Nadopunite odsječak koda:

```
class Solution{

    public static Predicate<List<List<Integer>>> allDigitsMatch(double exemplar){
        return  /* ovdje dodati rjesenje */
    }


    public static Predicate<List<List<Integer>>> allDigitsDefined(){
        return   /* ovdje dodati rjesenje */
    }

}
```

prema sljedećim zahtjevima:

- metoda `allDigitsMatch` vraća predikat koji omogućuje testiranje odgovara li zapis znamenki u listi znamenkama priloženog decimalnog broja.
- metoda `allDigitsDefined` vraća predikat koji omogućuje testiranje jesu li zapisu znamenki u listi zapisane sve uzastopne znamenke.
- čitavu klasu `Solution` zalijepiti u Edgar.

Primjer

```
List<List<Integer>> ulaz1= Arrays.asList(Arrays.asList(1,2),Arrays.asList(0,3),Arrays.asList(2,6)); // 3.26

boolean t11 = Solution.allDigitsMatch(3.266).test(ulaz1); // true
boolean t12 = Solution.allDigitsDefined().test(ulaz1); // true

List<List<Integer>> ulaz2=
Arrays.asList(Arrays.asList(1,2),Arrays.asList(0,3),Arrays.asList(4,6)); // 3.2**6
boolean t21 = Solution.allDigitsMatch(3.266).test(ulaz2); // true
boolean t22 = Solution.allDigitsDefined().test(ulaz2); // false
```

NAPOMENA: pretpostavite da su testovi takvi da su svi redni brojevi nenegativni i jedinstveni, te znamenke jednoznamenkaste.

Student's answer:

```
import java.util.*;
import java.util.function.Predicate;

class Solution{

    public static Predicate<List<List<Integer>>> allDigitsMatch(double exemplar){
        String number = String.valueOf(exemplar);
        char[] charDigits = number.toCharArray();
        List<Integer> digits = new ArrayList<>();
        for (char c : charDigits) {
            if(c != '.') {
                digits.add(Integer.parseInt(String.valueOf(c)));
            }
        }
        return lists -> {
            for (List<Integer> list : lists) {
                if(!digits.contains(list.get(1))) {
                    return false;
                }
            }
            return true;
        };
    }


    public static Predicate<List<List<Integer>>> allDigitsDefined(){
        return lists -> {
            Set<Integer> indexes = new TreeSet<>();
            for (List<Integer> list : lists) {
                indexes.add(list.get(0));
            }
            for (int i = 0; i < indexes.size(); i++) {
                if(!indexes.contains(i)) {
                    return false;
                }
            }
            return true;
        };
    }


    // this is from allDigitsMatch and it does... something...
    //        String number = String.valueOf(exemplar);
    //        char[] charDigits = number.toCharArray();
    //        List<Integer> digits = new ArrayList<>();
    //        for (char c : charDigits) {
    //            if(c != '.') {
    //                digits.add(Integer.parseInt(String.valueOf(c)));
    //            }
    //        }
    //        List<List<Integer>> distributed = new ArrayList<>();
    //        for (int i = 0; i < digits.size(); i++) {
    //            List<Integer> onePosition = List.of(i, digits.get(i));
    //            distributed.add(onePosition);
    //        }
    //        return lists -> {
    //            lists.sort(Comparator.comparing(o -> o.get(0)));
    //            for (int i = 0; i < lists.size(); i++) {
    //                if(!lists.get(i).equals(distributed.get(i))) {
    //                    return false;
    //                }
    //            }
    //            return true;
    //        };
}
```

Correct answer:

```
This test does not have "show solutions" option enabled.
```

Hint: Correct. Well done!

Exam results obtained during submission evaluation:

#	stdin	stdout	expected	stderr	percentage	hint	mode	correct?
1	functionality-allmatch	OK	OK	EdgarLibrary v0.11.1	50.00	Correct. Well done!	check elements order : false, case sensitive : false, ignore whitespace : true	true
2	functionality-alldefined	OK	OK	EdgarLibrary v0.11.1	50.00	Correct. Well done!	check elements order : false, case sensitive : false, ignore whitespace : true	true

Student's result | Correct result

Rerun student's code