ES112: Computing
Lab 05 — 15 October 2018

Topic: Lists and Sets

1. You are given two lists. List 1 contains ids of workers who are proficient in task 1. List 2 contains ids of workers who are proficient in task 2. The ids are alphanumeric. Due to some mistake by the clerk entering the data, the lists might contain duplicates.

   Remove the duplicates from both the lists and finally output a list containing the ids of the workers who are proficient in task 2 only and not in task 1. The output should be a sorted list.

   Sample input.

   ```
   list1 = ['BCE017','BIT049','BCE017']
   list2 = ['BCE017', 'BIT059', 'BIT049', 'BIT059']
   ```

   Expected output.

   ```
   ['BIT059']
   ```

   Function Signature

   ```
   def solve01(list1,list2):
   ```

   - **list1** is a list containing **strings**.
   - **list2** is a list containing **strings**
   - The expected return type is a **list** containing strings.

2. Alien Fred wants to destroy the Earth, but he forgot the password that activates the planet destroyer.

   You are given a **`str`** **`S`**. Fred remembers that the correct password can be obtained from **`S`** by erasing exactly one character.

   Return the number of different passwords Fred needs to try.

   Sample input

   ```
   "ABA"
   ```

   Expected output

   ```
   3
   ```

   The following three passwords are possible in this case: "BA", "AA", "AB".

   ```
   "ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
   ```

   Expected output

   ```
   1
   ```

   Explanation

   Regardless of which character we erase, we will always obtain the same string. Thus there is only one possible password: the string that consists of 49 'Z's.

   Function Signature

   ```
   def solve02(S):
   ```

   - **`S`** is a **`str`** and the expected return type is an **`int`**.

   Other Remarks

   The quotes showin the in sample input are only for clarity. The actual string represented by the variable **`S`** will not have the quotes.

   `S` will contain between 1 and 50 characters, inclusive and each character in **`S`** will be an uppercase English letter (**`'A'-'Z'`**).

3. There are **N** students who took part in the voting. Each student has a student ID using that they will vote. You are given **N** and a **list** of integers corresponding to the ID of the students who cast the votes.

   Help the officials to find out if any one has voted more than once, by giving them the total number of fraud votes.

   Each student is supposed to cast at most one vote only. So, if a student has cast more than one vote, all votes except one vote will be fraud votes.

   Sample input 1.

   ```
   N = 4
   a = [1, 2, 3, 2]
   ```

   Expected output 1.

   ```
   1
   ```

   Sample input 2.

   ```
   N = 3
   a = [1, 2, 3]
   ```

   Expected output 2.

   ```
   0
   ```

   Function Signature

   ```
   def solve03(N, a):
   ```

   - **N** is an integer and **a** is list of integers.
   - The expected return type is *int*.

4. There is an array, **`arr`** of n integers. There are also two disjoint sets, **`A`** and **`B`**, each containing m integers. Tintin likse all the integers in set **`A`** and dislikes all the integers in set **`B`**. His initial happiness is **`0`**. For each integer in the array i, if $i \in A$, his happiness increases by 1. If $i \in B$, it decreases by 1. Otherwise, the happiness does not change. Output Tintin's final happiness.

   Sample input

   ```
   arr = [5, 1, 3, 3, 2]
   A = [1, 3]
   B = [5, 7]
   ```

   Expected output

   ```
   1
   ```

   Explanation

   In the given array there are two elements,1 and 3 that he likes, so his happiness increases by 2 and since he doesn't like 5, it decreases by 1. Hence the happiness in the end is $+1$.

   Function Signature

   ```
    def solve04(arr, A, B):
   ```

   - **`arr`** is an array containing n integers.
   - **`A`** and **`B`** are arrays containing m integers.
   - The expected return type is int.

   Other Remarks.

   Since **`A`** and **`B`** are given to be sets, they will have no repeated elements. However, the array arr might contain duplicate elements. Ensure that for no element in **`arr`**, the happiness is counted twice!

5. Augustus and Beatrice play the following game. Augustus thinks of a secret integer number from **1** to **n**. Beatrice tries to guess the number by providing a set of integers. Augustus answers **YES** if his secret number exists in the provided set, or **NO**, if his number does not exist in the provided set. Then after a few questions Beatrice, totally confused, asks you to help her determine Augustus's secret number.

   Given the positive integer **n** as input, followed by a **list** called **guesses** consisting of tuples of size two, the first element of which consists of Beatrice's guessed set, represented as a space-separated string of integers and the corresponding response from gustus. Provide a list of all the remaining possible secret numbers, in ascending order, separated by a space.

   Sample input

   ```
   n = 10
   guesses = [("1 2 3 4 5", "YES"), ("2 4 6 8 10", "NO")]
   ```

   Expected output

   ```
   "1 3 5"
   ```

   Function Signature

   ```
    def solve05(n,guesses):
   ```

   - **n** is an **int** and **guesses** is a list of tuples of strings of size two.
   - The expected return type is a **str**.

6. Recall that we discussed this problem in the second class of this course, which we are now prepared to solve. This is the problem statement of *Saving the Universe*, the first problem of Google Code Jam back in 2008. Recall the story:

   The urban legend goes that if you go to the Google homepage and search for "Google", the universe will implode. We have a secret to share... It is true! Please don't try it, or tell anyone. All right, maybe not. We are just kidding.

   The same is not true for a universe far far away. In that universe, if you search on any search engine for that search engine's name, the universe does implode!

   To combat this, people came up with an interesting solution. All queries are pooled together. They are passed to a central system that decides which query goes to which search engine. The central system sends a series of queries to one search engine, and can switch to another at any time. Queries must be processed in the order they're received. The central system must never send a query to a search engine whose name matches the query. In order to reduce costs, the number of switches should be minimized.

   Your task is to tell us how many times the central system will have to switch between search engines, assuming that we program it optimally. Assume you are given as input two integers **`S, Q`**, the number of search engines and queries, respectively; and two lists consisting of the names of the search engines and the names of the queries. You are to return the smallest number of switches.

   Sample input

   ```
   S = 5, Q = 10
   engines = ["Yeehaw", "NSM", "Dont Ask", "B9", "Googol"]
   queries = ["Yeehaw", "Yeehaw", "Googol", "B9", "Googol", "NSM", "B9", "NSM", "Dont
   Ask", "Googol"]
   ```

   Expected output

   ```
   1
   ```

   Function Signature

   ```
   def solve06(S,Q,engines,queries):
   ```

   - **`S`** and **`Q`** are integers while **`engines`** and **`queries`** are lists of strings.
   - The expected return type is an **`int`**.

   Other Remarks

   Each search engine name is no more than one hundred characters long and contains only uppercase letters, lowercase letters, spaces, and numbers. There will not be two search engines with the same name.

7. A permutation of length **n** is an array of size n consisting of **n** *distinct* integers in the range **[1,n]**. For example, (3, 2, 4, 1) is a permutation of length 4, but (3, 3, 1, 4) and (2, 3, 4, 5) are not, as (3, 3, 1, 4) contains duplicate elements, and (2, 3, 4, 5) contains elements not in range [1,4].

   A permutation p of length n is **good** if and only if for any $1 \leqslant i \leqslant n, p[i] \neq i$.

   Given **n**, your task is to find the *lexicographically smallest* good permutation p.

   Example of "lexicographically smaller" for some permutations with **n = 4**:

$$(2, 3, 1, 4) < (2, 3, 4, 1) < (3, 4, 1, 2)$$

   The lexicographically smallest permutation is, of course, (1, 2, ..., n), though this one is not good.

   Sample input 1.

```
2
```

   Expected output.

```
[2, 1]
```

   Explanation
   The only good permutation of length **2** is **(2, 1)**.

   Sample input 2.

```
3
```

   Expected output.

```
[2, 3, 1]
```

   Function Signature

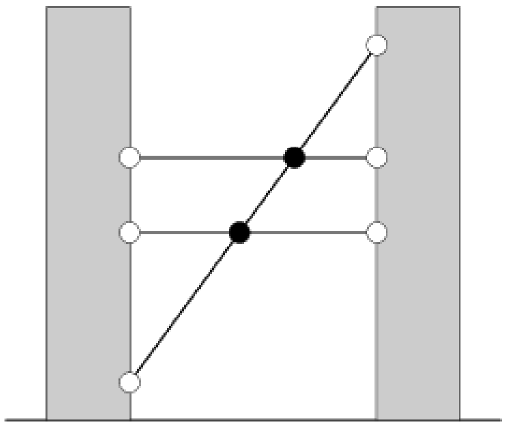
```
def solve07(n):
```

   - n is of type **int**.
   - The expected return type is **list**.

8. (Based on *Rope Intranet*, Google Code Jam 2010 Round 1C.)

   A company is located in two very tall buildings. The company intranet connecting the buildings consists of many wires, each connecting a window on the first building to a window on the second building.

   You are looking at those buildings from the side, so that one of the buildings is to the left and one is to the right. The windows on the left building are seen as points on its right wall, and the windows on the right building are seen as points on its left wall. Wires are straight segments connecting a window on the left building to a window on the right building.

   You've noticed that no two wires share an endpoint (in other words, there's at most one wire going out of each window). However, from your viewpoint, some of the wires intersect midway. You've also noticed that exactly two wires meet at each intersection point.

   On the above picture, the intersection points are the black circles, while the windows are the white circles. You are given as input a list of **n** tuples describing the left and right endpoints of the wires - these are integers describing the height of the windows on the left and the right buildings, respectively. Return the number of intersection points that you see.

   Sample input

   ```
   wires = [(1,10),(5,5),(7,7)]
   ```

   Expected output

   ```
   2
   ```

   Function Signature

   ```
   def solve08(wires):
   ```

   - The input is a **list** of tuples of size two.
   - The expected return type is an **int**.

9. (Based on *Watersheds*, Google Code Jam 2009 Qualifiers.)

   Geologists sometimes divide an area of land into different regions based on where rainfall flows down to. These regions are called drainage basins.

   Given an elevation map (a 2-dimensional array of altitudes), label the map such that locations in the same drainage basin have the same label, subject to the following rules.

   - From each cell, water flows down to at most one of its 4 neighboring cells.
   - For each cell, if none of its 4 neighboring cells has a lower altitude than the current cell's, then the water does not flow, and the current cell is called a sink.
   - Otherwise, water flows from the current cell to the neighbor with the lowest altitude.
   - In case of a tie, water will choose the first direction with the lowest altitude from this list: North, West, East, South.

   Every cell that drains directly or indirectly to the same sink is part of the same drainage basin. Each basin is labeled by a unique lower-case letter, in such a way that, when the rows of the map are concatenated from top to bottom, the resulting string is lexicographically smallest. (In particular, the basin of the most North-Western cell is always labeled **'a'**.)

   You are given the elevation map as a two-dimensional list of integers called **elevations**. Return the map of labels as a two-dimensional list of strings of the same dimensions as the elevation map.

   Note that **elevations[0][0]** denotes the north-western cell and **elevations[i][j]** denotes the cell corresponding to row i and column j.

   Sample input 1.

```
elevations = [[9, 6, 3], [5, 9, 6], [3, 5, 9]]
```

   Expected output 1.

```
[['a', 'b', 'b'], ['a', 'a', 'b'], ['a', 'a', 'a']]
```

   Sample input 2.

```
elevations = [[0, 1, 2, 3, 4, 5, 6, 7, 8, 7]]
```

   Expected output 2.

```
[['a','a','a','a','a','a','a','a','a','b']]
```

   Function Signature

```
def solve09(elevations):
```

   - The input is a two-dimensional list of integers.
   - The expected output is a two-dimensional list of strings.

   Other Remarks

   There will be at most 26 basins to allow for labeling with lower-case English letters only.

10. (Based on *Candies* at HackerRank.)

Alice is a kindergarten teacher. She wants to give some candies to the children in her class. All the children sit in a line and each of them has a rating score according to his or her performance in the class. Alice wants to give at least one candy to each child. If two children sit next to each other, then the one with the higher rating must get more candies. Alice wants to minimize the total number of candies she must buy.

For example, assume her students' ratings are **`[4, 6, 4, 5, 6, 2]`**. She gives the students candy in the following minimal amounts: **`[1, 2, 1, 2, 3, 1]`**. She must buy a minimum of **`10`** candies.

Given a **`list`** of integers **`ratings`** as input, return the smallest number of candies that Alice must buy.

Sample input

```
[1,2,2]
```

Expected output

```
4
```

Explanation.

Here 1, 2, 2 is the rating. Note that when two children have equal rating, they are allowed to have different number of candies. Hence optimal distribution will be 1, 2, 1.

Sample input

```
[2, 4, 2, 6, 1, 7, 8, 9, 2, 1]
```

Expected output

```
19
```

Explanation.

An optimal distribution would be: 1,2,1,2,1,2,3,4,2,1.

Function Signature

```
def solve10(ratings):
```

- **`ratings`** is a **`list`** of integers.
- The expected return type is an **`int`**.