COL100 Assignment 7
2^{nd} Semester Semester : 2021-2022

Deadline: 11:59 pm, 22 May, 2022

General Instructions

You should attempt this assignment without taking help from your peers or referring to online resources except for documentation (we will perform a **plagiarism check** amongst all submissions). Any violation of above will be considered a breach of the honor code, and the consequences would range from **zero marks** in the assignment to a **disciplinary committee action**.

Submission Instructions

1. **If you are solving the i^{th} question, code in a file named `<EntryNo>-qi.py`.** For eg. solution code for the 2^{nd} question goes inside a file named `2021MCS2168-q2.py` , **Please ensure this point, otherwise 0 will be given.**

2. You may submit the lab question mentioned in the assignement as "In Lab Component". The question evaluation for a lab will be done in the lab only and evaluations not done for the in-lab component for a student in his/her lab slot will be marked 0. It is your duty to get them evaluated.

3. **Submit your code in a `.zip` file named in the format `<EntryNo>.zip`. Make sure that when we run unzip `<EntryNo>.zip`, a folder `<EntryNo>` should be produced in the current working directory.** For eg. if your entry number is `2021CS5XXXX`, then your zip file would be `2021CS5XXXX.zip` and upon unzipping, it should produce a folder `2021CS5XXXX` containing files `<EntryNo>-q1.py`, `<EntryNo>-q2.py`, `<EntryNo>-q3.py` and so on. We have provided a zip that contain the skeleton code with this directory structure and similar naming convention.**Please ensure this point, otherwise 0 will be given.**

4. Your submissions will be **auto-graded**. Make sure that your code follows the specifications (including directory structure, input/output, importing libraries, submission `.zip` file) of the assignment precisely.

Some Clarifications

1. Every problem description is followed by some examples showing how exactly input and output is being expected. Please refer to them for more clarity.**Please ensure this point, otherwise 0 will be given.**

2. All the inputs/outputs are **integers** unless explicitly mentioned.

3. If you still have any more doubts, feel free to shoot them at Piazza.

4. You can use lists to store the data in the questions.

5. You can use common inbuilt functions like append/split/len/etc freely. Import statement already given in q3 skeleton code - "from typing import List" is allowed and **do not import anything except this in q3.** Also, **Do not import anything in q1 and q2**. **Do not use count() inbuilt function**.

6. **Write your code only in skeleton code provided. Do not change function names, argument order, and input-output statements , etc already in the skeleton code. Do not comment the main function. Just implement the function and submit the code**

1 Stairway to Heaven (10 marks)(In Lab Component)

Mr. Page has found the stairway to heaven. The number of steps to go to heaven is N. He is currently at the 0^{th} stair and wishes to reach the N^{th} stair. Mr. Page can take 1 or 2, or 3 steps in a single jump. Mr. Page is not a mathematician and needs your help determining the number of ways in which he can reach heaven.

For example, if $N = 9$ then some of the possible jump combinations are: $(3, 3, 3), (2, 2, 2, 3), (3, 3, 2, 1)$.... For example, $(3, 3, 3)$ is a valid jump combination because $3 + 3 + 3 = 9$ and each jump covers 1, 2, or 3 steps only. Similarly, for $(2, 2, 2, 1)$, we have $2 + 2 + 2 + 3 = 9$, so it is a valid combination. The total number of valid combinations of jumps will be 149 for the input $N = 9$, and Mr. Page only needs this number (not the list of all jump combinations).

Your solution should run in linear time - O(N). Here N, is the number of steps to reach heaven. **You have to write a short description (2-3 lines) of why your code has linear time complexity, write it inside the comments in the skeleton code.**

We have provided a skeleton code which takes input; type cast the input into int; call the **function `heaven()`** and print the return value of the function `heaven()`. You have to only implement the function `heaven()` and **return the output as integer — you must not print anything in the function**.

Input: The first line (only input line) of input will contain the value of N (only integer (int type); follow the skeleton code.).

Return Output : Only return the total number of combinations of jumps (only integer (int type)).

Some points to note:

1. Any algorithm of time complexity more than O(N) will receive 0. As the stairs go to heaven, they might have large values.
2. Combinations like $(1, 3)$ and $(3, 1)$ are considered distinct and are counted as 2 while combination like $(1, 1)$ and $(1, 1)$ are same and considered as 1 .

Example 1:
`INPUT`:

```
300
```

`OUTPUT`:

```
153502876143597386718435065670236352689242811730518018615665246091844610209903 67
```

Example 2:
`INPUT`:

```
4
```

`OUTPUT`:

```
7
```

`EXPLANATION`:
Possible Combinations are [(3,1),(1,3),(2,2),(2,1,1),(1,2,1),(1,1,2),(1,1,1)]. Total 7

2 Sorting (10 marks)

You have been given a string that can contain digits (base 10) or letters('a'-'z' or 'A'-'Z') only. You have to arrange the characters of this string in such a way that all occurrences of digits (0-9) which occur the odd number of times are moved to the front of the string in ascending order. The rest of the characters in the string are to be preserved.

We have provided a skeleton code which takes input; call the **function `sortings ()`** and print the returned output (which you have returned from function `sortings ()`). You have to only implement the function `sortings()` and **return your output without printing anything in the function.**

Input: The first line of input (only input line) will contain string (can only contain digits (base 10) or letters, follow skeleton code.).

Return Output : Only "return" the string arranged in the manner mentioned above. (only string type and string contains only digits (base 10) or letters).

Some points to note:

1. In this question, we will be checking ONLY your output, so make sure to return correct format. Any output format mismatch will result in 0.
2. Characters of strings can be case sensitive like "A" and "a" are different.

Constraint: Length of string can be 0 ("") and can go as large as you can think.

Example 1:
`INPUT`:

```
81454aDc5445bd
```

`OUTPUT`:

```
1555844aDc44bd
```

`EXPLANATION`:
Digits occurring the odd number of times are = 8, 1, 5.
Also, 5 occurs three times.
Remaining characters in the string are `44aDc44bd`.
So string returned by `sortings()` is `1555844aDc44bd`

Example 2:
`INPUT`:

```
322fcd223
```

`OUTPUT`:

```
322fcd223
```

`EXPLANATION`:
Here, digits occurring odd number of times are none. So the character left is the same as the original.

3 Sorting a list (25 marks)

You have been given a list of strings and each string can contain digits (base 10) or letters only. For each string, you have to arrange the character of them (each string) in the same manner as mentioned as in Question 2. After this, for the resulting list of strings, you have to sort the list as follows - for each string, you have to extract all digits starting from index 0 until any non-digit character ("a"-"z" or "A"-"Z") comes or the end of a string is encountered. Compare the extracted numerical value of one string with extracted numerical values of other strings to sort the list in ascending order.

For example, if a string output of question 2 is "1112355744hkj887"; then extracted numerical value will be 1112355744.

We have provided a skeleton code which takes input from you; call the **function "sortings"** and print the returned output (which you have returned from function "sortings"). You have to write your code, in function "sortings" only and **"return" your output and do not need to print anything in function.**

Input: The first line of input will contain the number of strings in the list (int type). From the second line onwards, each line will contain a string of the list. (str type, follow skeleton code.).

Return Output : Only "return" the **list of strings** in the sorted manner as explained in the question. (Only return list , follow skeleton code).

Some important points to note:

1. In this question, we will be checking ONLY your Output, so make sure to return the correct format. Any output format mismatch will result in 0.
2. Follow skeleton code, otherwise you can loose marks. You can make your own function which can be called from function "sortings"
3. If for two strings, numerical values are the same, then either string can be in front of the other (order does not matter for these two).
4. If no numerical value can be extracted from the string, then it is to be put at the last of list because it's equivalent numerical value will be **None** and not 0.
5. If length of string is 0 (""), then it's equivalent numerical value will be **None** and not 0.

Constraint: Length of string can be 0 ("") and can go as large as you can think. Length of list can be 0 that is `([])` and can go as large as you can think. Use of None is allowed.

Example 1:
INPUT:

```
4
44332211
4412355bac889966
abcdef5588447755
44332211abc
```

OUTPUT:

```
['1234455bac889966', '44332211', '44332211abc', 'abcdef5588447755']
```

EXPLANATION:

1. The given list is ['44332211', '4412355bac889966', 'abcdef5588447755', '44332211abc'].
2. After calling Question 2 for each string we get ['44332211', '1234455bac889966', 'abcdef5588447755', '44332211abc'].
3. Extracted numerical value are [44332211, 1234455, None, 44332211].
4. Sorting these numerical value we get [1234455, 44332211, 44332211, None].
5. Thus, Output list will be ['1234455bac889966', '44332211', '44332211abc', 'abcdef5588447755']

Example 2:
INPUT:

```
3
12334abc566
7891010efg
00000hgf0
```

OUTPUT:

```
['00000hgf0', '124533abc66', '7891010efg']
```