CSE 101 - Introduction to Programming
Assignment 2
February 2022

GENERAL INSTRUCTIONS

1. In this assignment, you need to write the code for every problem in different '.py' files.
2. You are only allowed to use the basic modules available in Python. You cannot use any external modules, libraries, or APIs.
3. You will create a document, citing all the online and offline resources, that you have used, in completing this assignment.
4. Your code will be checked for plagiarism against the code of your classmates as well as some sample codes available online. The institution's policies will strictly apply.
5. This time, questions will be added in phases, i.e, further questions would be added. You will be notified regarding this.
6. Start early. Resolve all your doubts with the TAs, *2 days* before the deadline.

SUBMISSION INSTRUCTIONS

1. Your code files must be renamed as A2_2021xxx_{Problem_no}.py. For example, A2_2021001_1.py, would be the python file for problem 1 for roll no. 2021001.
2. Your document with references to resources used must be in pdf format and named A2_Resources_2021xxx.pdf
3. All these files (code files and the reference pdf file) must be put in a zipped folder named A2_2021xxx.zip
4. The zip file containing everything relevant to this assignment must be submitted before the deadline on google classroom.

PROBLEMS

Q 1. Word and Words!!

For this problem, you have to write a program that performs some operations on a text file. The program offers the user a choice to select one operation at a time. The operations are listed below:

1. **Display specific Word Count**: The user is prompted to input the word whose count is required. If the word exists in the text file, print its count and if not then print 'Word does not exist'.
2. **Display Unique Words**: For this choice, you just print all the unique words present in the text. (Note that a word is any sequence of characters/digits/alphabets enclosed between a space)
3. **Display all Word Counts**: For this option, you have to print all the unique words and their corresponding word counts. (*Hint: Use a dictionary!*)
4. **Replace Word**: You prompt the user to input two words, say Word1 and Word2. All the occurrences of Word1 in the text file are replaced by Word2.
5. **Quit**: The program ends.

The program is menu-driven (see figure below).

```
----------------------------------------------------------------------
Enter your choice:
1. Display specific Word Count
2. Display all Unique Words
3. Display all Word Counts
4. Replace word
5. Quit
█
```

If the user chooses option 1, then a sample output is shown below.

```
----------------------------------------------------------------------
Enter your choice:
1. Display specific Word Count
2. Display all Unique Words
3. Display all Word Counts
4. Replace word
5. Quit
1
Enter word: the
Word Count:  11
```

For option 2, print all the unique words present in the text separated by a semicolon (;)

```
-------------------------------------------------------------------------
Enter your choice:
1. Display specific Word Count
2. Display all Unique Words
3. Display all Word Counts
4. Replace word
5. Quit
2
Unique Words:
 ; face. ; before ; multiply ; first ; The ; has ; cold ; back ; these ; is ; likely ; Over ; It'
s ; body ; now ; that ; contaminated ; (within ; COVID-19. ; because ; membrane ; from ; possible ; m
outh, ; nasal ; surface ; territories ; other ; there, ; tissue. ; 4,850,000 ; East ; transfer ; (10/
11/2021), ; your ; you ; or ; throat. ; common ; tissues. ; Middle ; released ; coughs, ; cause ; vir
uses ; COVID-19 ; a ; on ; They ; get ; continents ; illness ; contact ; shaking ; the ; spikes ; As
 ; hands ; hands. ; inhale ; hands) ; then ; may ; crown-like ; number ; sings ; than ; 1. ; through ;
 examples ; Wuhan, ; spread: ; was ; touching ; 3. ; talks, ; 6 ; into ; But ; syndrome ; (touching,
 ; changes ; lung ; and ; enters ; It ; washing ; this ; daily. ; air ; Severe ; coronaviruses ; 2019.
 ; are ; can ; died. ; be ; attaches ; eyes, ; From ; all ; when ; infected ; droplets ; 196,910,000
 ; called ; been ; sneezes, ; - ; airborne ; in ; infected. ; face). ; breathes ; virus ; feet). ; per
son ; near ; to ; reported ; considered ; world ; 2. ; family ; China ; virus. ; moves ; unlikely. ;
people ; (directly ; cells ; begins ; acute ; Some ; Coronaviruses ; countries ; passages ; "corona"
 ; new ; nose ; (SARS), ; (MERS) ; cases ; spread ; continents. ; more ; after ; with ; respiratory ;
192 ; coronavirus ; close ; thought ; You ; since ; of ; strain ; an ; travels ; also ; writing ; Dec
ember ; it's ; humans. ; if ; eyes ; have ; mucous ; droplets.
```

For choice 3, the following is the output (cropped version):

```
---------------------------------------------
Enter your choice:
1. Display specific Word Count
2. Display all Unique Words
3. Display all Word Counts
4. Replace word
5. Quit
3
Word Counts:
Two :  1
members :  1
of :  4
the :  11
1984 :  1
class :  2
Jefferson :  1
High :  1
```

For choice 4, the user is required to enter 2 words, Word1 and Word2. All occurrences of Word1 are replaced by Word2 in the file.

```
--------------------------------------------
Enter your choice:
1. Display specific Word Count
2. Display all Unique Words
3. Display all Word Counts
4. Replace word
5. Quit
4
Enter word to be replaced: the
Enter word that will replace the: program
Replaced successfully!
```

- The input text file will be provided.
- Consider the following example regarding Option 2 and 3:
  Let's say that the text file has the following contents: "the the is a break the if if"
  Then the unique words in the text file are ["the", "is", "a", "break", "if"] *(and not ["is", "a", "break"]).*
  So the output for Option 2 will be: the ; is ; a ; break ; if
  For Option 3 the output becomes:
  the : 3
  is : 1
  a: 1
  break : 1
  if : 2
- For Option 4, you have to rewrite the contents in the input file itself by replacing Word1 with Word2. (*No new file should be created*). The options can be chosen in any order and hence the results should be in accordance with the *updates* made by the user.
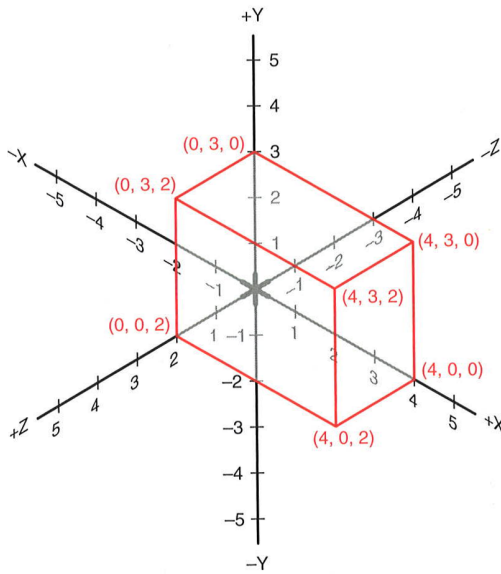- In the final submission make sure that the input file is unaltered.

Q2. Transformations!!

In 3D graphics software like Blender, Maya, etc., we are able to create objects and perform some transformation operations on them. Making software like so is beyond the scope of this course but we can make a simple one that works on the console. Your task is to write a program that is able to create a 3D shape from given vertices and is able to perform 3 transformation operations which are Scaling, Translation, and Rotation.

Creating the 3D shape
The program takes as input three lists x, y, and z each of length n where ($x[i]$, $y[i]$, $z[i]$) represent the ith vertex of the 3D Shape.

Example:

For the above cuboid
x = [0, 0, 0, 0, 4, 4, 4, 4]
y = [0, 0, 3, 3, 3, 0, 0, 3]
z = [0, 2, 2, 0, 0, 0, 2, 2]

Transformation
Any transformation T represented by a matrix (list of lists) can be applied on a point (x, y, z) to get the transformed point (x′, y′, z′).

A generic T can be represented as -

$$
T = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

So, applying T on point (x,y,z) is done using -

$$
\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
$$

There are 3 types of transformations as mentioned earlier and each type of transformation has it's own T matrix representation.

1. Scaling
   For scaling, to scale the point by sx , sy and sz ; T here is -

$$
T = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

2. Translating
   For translating a point by tx, ty, and tz ; T here, is -

$$
T = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

3. Rotation
   The rotation can happen along 3 axes viz., x, y and z so each of these have different T matrix represented by $\text{rotate}_{\text{axes}}(\phi)$ where ϕ is the angle by which you wish to rotate the point.

$$
\underbrace{\begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{rotate}_z(\phi)}, \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{rotate}_x(\phi)}, \underbrace{\begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{rotate}_y(\phi)}
$$

Input:
The first line will contain **n** which is the number of vertices the 3D shape has
Next line will input the space-separated list **x**
Next line will input the space-separated list **y**
Next line will input the space-separated list **z**
Next line will input the number of transformation queries to apply which is **q**
Next **q** lines will input a query

Each query looks like -

For scaling, the query looks like
S sx sy sz
where, **S** represents scaling and **sx, sy** and **sz** represent the amount of scale in each axes

For translating, the query looks like
T tx ty tz
where, **T** represents translation and **tx, ty** and **tz** represent the amount of translation in each axes

For rotating, the query looks like
R x ϕ
where, **R** represents rotation and **x** represents the axis of rotation and ϕ represents the angle by which to rotate. Similarly, for y-axis we write **R y ϕ** and for z axis we write **R z ϕ**

Output:
Print the final transformed 3 lists x, y and z

Also, store these lists in a text file in the same format as input and output.

Q3. Conversion of numbers !!.

As we all know, we can do numbers representation in the different radix. On Planet Earth, humans use numbers with radix 10, and the computer uses numbers with radix two, etc. Scientists at NASA recently found that creatures living there may use different radix to represent a number on several other planets. For example, Creatures living on the Huru-Huru planet uses hexadecimal radix. On the other hand, creatures of the Zora planet uses octal representation. Scientists want to communicate with these creatures from a different world, so we need to convert numbers from one radix to another. For this task, NASA hired you. As a programmer at NASA, you are asked to create a menu-driven program for the following operations. For each process, you must make appropriate functions. For each of the following operations, you must ask a number represented in string as an

input and give the result in the other radix. You must handle it wisely if the user does not enter a number with the specified radix.

1) Convert decimal to binary and vice-versa

2) Convert decimal to hexadecimal and vice-versa

3) Convert decimal to octal and vice-versa.

4) Convert binary to hexadecimal and vice-versa.

5) Convert binary to octal and vice-versa.

6) Convert hexadecimal to octal and vice-versa.

7) Convert number with radix A to radix B. Here A,B <= 36. In this case, you must take A,B as input.

Type 1-6 are basic and you already know examples.

Example of Type 7) :

$(GF12)_{18} = (98192)_{10}$ (here, A = 18 , B = 10)

$(CA91)_{13} = (5C83)_{17}$ (here, A = 13, B = 17)
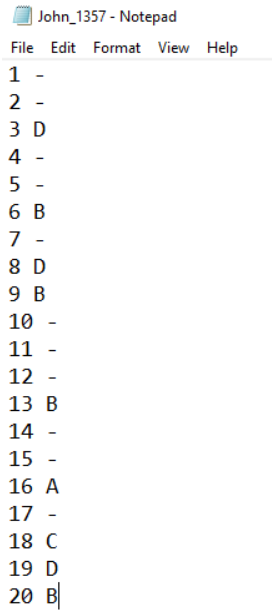
Q4. Final Report!!

The National Testing Agency will not be able to hold this year's JEE paper. So, they have asked you to conduct the JEE paper but before that, they want to test you if you can write a program that can simulate a JEE-type environment.

The question paper has a total of 20 MCQ-type questions.

There are two folders representing the two high-level parties of the system which are "Student" and "Admin".

In the "Student" folder there will be text files with the naming convention "StudentName_StudentNo.txt". Each student has their own text file. Each text file represents the student's submission to the questions.

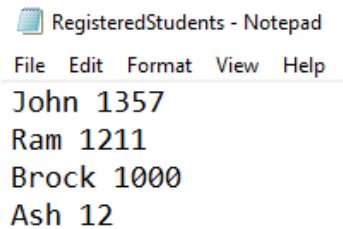An example text file for the Student Name - John with Student Number - 1357 is

```
John_1357 - Notepad
File  Edit  Format  View  Help
1  -
2  -
3  D
4  -
5  -
6  B
7  -
8  D
9  B
10  -
11  -
12  -
13  B
14  -
15  -
16  A
17  -
18  C
19  D
20  B
```

Where, "-" represents that the student didn't attempt that question

There will be a "RegisteredStudents.txt" in the "Admin" folder.

Example of a RegisteredStudents.txt -

```
RegisteredStudents - Notepad
File  Edit  Format  View  Help
John  1357
Ram  1211
Brock  1000
Ash  12
```

Where, each line represents <StudentName> <StudentNo>

In the "Admin" folder there is also an "AnswerKey.txt" file which is the answer key to the question paper. Using this answer key, you need to evaluate the submissions of each student. And, finally report a "FinalReport.txt" which has the name, number, and score of each student.

Example of AnswerKey.txt -

AnswerKey - Notepad

File Edit Format View Help

```
1  B
2  B
3  D
4  A
5  B
6  B
7  A
8  C
9  C
10  B
11  A
12  A
13  C
14  A
15  B
16  D
17  B
18  B
19  C
20  D
```

Example of FinalReport.txt -

FinalReport - Notepad

File Edit Format View

```
John 1357 1
Ram 1211 60
Brock 1000 52
Ash 12 -2
```

Where, each line represents <StudentName> <StudentNo> <StudentScore>

The scores are evaluated as follows -

- If the student's answer matches the answer in the answer key they are awarded +4 marks
- If the student answered the question and it does not match the answer in the answer key they are awarded -1 marks
- If the student didn't answer the question they are awarded 0 marks.

Your task is to write a code that evaluates each student's submission and finally create a FinalReport.txt

Q5. Music Theory!

Most of the music cultures across the globe have the most fundamental similarity. This similarity is called **"Note".** A note is a fundamental unit of music that is used to create melodies and riffs.

Typically, you must have heard that music consists of seven notes (or the "**Sapt Suras**"). But in reality, we have 12 notes. 7 of them are the regular notes and 5 of the extra notes which are called sharp or flats(you don't need to worry about the theory behind it).

In simpler terms, If you have ever seen a piano, it consists of a block of 7 white keys and 5 black keys, and this pattern is repeated (also called an octave, or "**saptak**" in Hindi, although it does contain 12 notes!!)

The notes are typically denoted as

C C# D D# E F F# G G# A A# B C'

The **C' is the same note C, but in a higher octave (i.e, its frequency is double than the previous C !)**

Let's define some terms and jargons!!:-
- A **whole step**, denoted as **W,** is the jump from one note to the **3rd** note (i.e, a jump from C to D, leaving C# in between.) Remember these are not the random rules that I've made!!
- A half step, denoted as **H,** is the jump from the one note to the immediate next (i.e, from D to D# will be called a Half Step).
- A **Key** is the group of notes following some protocols (yes we have protocols in music as well!). The key is generally denoted by what is called the **ROOT note.** For example, A key of **'C'** has the root node as C.
- A **Scale** is a group of notes selected from the 12 fundamentals notes with a root note mentioned (For example, a C scale). A scale has two popular variants called **1. Major Scale** and, **2. Minor Scale**

→ To create a major scale based upon a root note, we follow the rule:

Root W W H W W W H

i.e., if we have to form a major scale with root note **C**, we will jump a whole step (**W**) to D, then another W to **E** (from D), Then half step (**H**) to **F** (from E**),** then another W to **G,** then W to **A,** Then W to **B**, and finally to **C'** (after half step from B**).**

So the notes of the major scale in C (i.e, with root node C) are **C, D, E, F, G, A, B, C'**

→ To create a minor scale based upon a root note, we follow the rule:

Root W H W W H W W

I.e, if we have to form a minor scale with root note C, we will follow the jumps (in W or H) and the notes we will be getting are: **C D D# F G G# A# C'**

Based upon this logic, You have to perform this task:-

First, you will be creating a function **noteCreate().** This function should create separate files called *'scaleMajor.txt'* and *'scaleMinor.txt'*. 'scaleMajor.txt' will contain notes of the major scale, for each of the 12 notes in separate lines. Similarly, 'scaleMinor.txt' will contain notes of minor scale for each of the 12 notes in separate lines. After having written this information, you must close the file.

You will be creating two functions called **majorNotes(root)** and **minorNotes(root).** Both of these functions accept a root note as a parameter (upper case). Then these will return the corresponding notes in the key of that root note.

You will be accepting the user's input for the root note and the type of scale ("Major" or "Minor") in separate lines.

Based upon the type of scale, you will be calling *majorNotes(root)* or *minorNotes(root).* Both majorNotes() and minorNotes() must be **reading the files** *scaleMajor.txt* and *scaleMinor.txt,* respectively depending upon the type of scale being asked by the user, and return the space-separated notes present in that scale.

```
Enter the root note: C
Enter the type of scale (Major/Minor): Major
The Major scale in the key of C is:
C D E F G A B C'
```

```
Enter the root note: C
Enter the type of scale (Major/Minor): Minor
The Minor scale in the key of C is:
C D D# F G G# A# C'
```

NOTE- File handling is mandatory for this program. All the steps for reading and writing to the files must be followed as has been mentioned.

Q6.

Given a 2d square matrix of size NxN, you are asked to traverse over matrix and print element present in the matrix in the order it has been traversed. You must first ask the user to give matrix input then create a menu for specified types of traversal, then ask the user to choose one of them and print the required traversal. You must make your code generalize so that it works properly for any given N.

1) Normal traversal(from left to right for each row)
2) Alternating traversal (first left to right for the first row, then right to left for the second row, then left to right for the third row, and so on.)
3) Spiral traversal from outer to inwards
4) Boundary traversal.(First, traverse the upper boundary from left to right, then right boundary from up to down, then bottom boundary from right to left and at last from left boundary from down to up)
5) Diagonal traversal from right to left
6) Diagonal traversal from left to right.

Sample case:

Consider matrix with N = 4:

```
1  2  3  4

5  6  7  8

9  10 11 12

13 14 15 16
```

Normal Traversal : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Alternating traversal : 1 2 3 4 8 7 6 5 9 10 11 12 16 15 14 13

Spiral Traversal : 1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10

Boundary traversal : 1 2 3 4 8 12 16 15 14 13 9 5

Diagonal traversal from right to left : 1 2 5 3 6 9 4 7 10 13 8 11 14 12 15 16

Diagonal traversal from left to right : 4 3 8 2 7 12 1 6 11 16 5 10 15 9 14 13

BONUS PROBLEMS

B1 Generate and Test!
Since you have been working on the fundamentals as well as, have learned new data structures and file handling, Dr. Jalote wants to test whether you are able to create a program that handles most of the concepts that you can apply.
For this specific task, you need to create 2 python files. In the first file called '*cases.py*', you need to create at least two mandatory functions-

```python
def function1():
    ## name of the function could be changed
    pass

def generateData():

    ##Must call function1
    ## inside it

    pass
```

   In **function1()**, you need to define any computational task of your choice. It should return some result(s).
In **generateData()**, you need to accept the user input for the number of input-output pairs, they would like to generate(say **N**). For this particular task, you have either of these two options:
   ● You can ask the user for inputs, N number of times. These inputs would have to be stored in N different text files. For each of the inputs, you need to call **function1()**. The result of function1() must be stored in another text file. Remember this output file must correspond

to the input being provided (Hint: you can add numbers to the input and output file names, respectively)

- You can use random() function to generate random data. After that, all the steps remain similar to the previous one.

In the second file called '*test.py*', you need to mandatorily use '*cases.py*'. And two mandatory functions.

```
##This file must use 'cases.py'
## as well as the function, generateData()
## inside it

def function2():
    ##This function is similar
    ## to function 1 but
    ## must be implemented differently
    pass




def testing():
    ##This function must use
    ## input output text file pairs generated earlier
    ## to read the data and test them on the
    ##results from function2
```

The function, *testing()* must read the inputs from the text files generated by *generateData()*. Then it must call *function2()*, to perform the operation on the input read from text file.

The result of *function2()*, must be matched to the output (which again, has to be read from the output text file), corresponding to the correct input. If all such outputs have been matched successfully, It should return "SUCCESS", Otherwise, "FAILED".

NOTE1: you are free to customize all the functions, with different names and number of parameters. You are also free to return any number or type of values from those functions. Every input passed to the functions, *function1()* and *function2()* must have been read from text files.
NOTE2: The function implementation in function1() is also left on your choice. function2() must implement the same operation as function1 but, differently (for example, if function1 uses for loop for iteration, function2() could use other iterative strategies).

Also, two skeleton files 'cases.py' and 'test.py', would be shared with the students, and they can edit them too.

NOTE:
All the related and relevant files have to be stored in a single folder named "**A2_2021XXX_B1**". The ".py" file specific to this question has to be present in this folder only, with the naming convention mentioned earlier.

B2.
Doja Dog and DJ Snack are 2 very popular, rich singers with many grammys. Moreover they

both win a grammy every year. They both love their fans and want to buy as many skyscrapers as they can for the fans to stay in the city of FanVille.

For every skyscraper of height h Doja Dog buys, it gives her a fan reputation of h * grammy. Here grammy is the number of grammys she has at the time she buys the skyscraper. In a similar way, reputation rises for DJ Snack when he buys a skyscraper.

Since the singing industry is competitive, Doja Dog and DJ Snack compete to get as much reputation as they can. So they both decide to buy the skyscraper with the maximum height available. But due to some weird laws in Fanville, only one skyscraper can be bought in a year. Hence to keep things fair, they both decide to buy skyscrapers on alternate years with Doja Dog going first.

You, as a fan of both Doja Dog and DJ Snack, want to find out who will have more fan reputation when all the skyscrapers are sold out. You also want to show the skyscrapers owned by each of the singers. As a programming student you decide to write a python program to find this out.

The initial number of Grammys with Doja Dog and DJ Snack are P, Q respectively.

The FanVille city is M units tall and has N skyscrapers. If M = 6 and N = 7, a possible layout for FanVille can be:

0000000

0000100

0010100

0110111

1110111

1111111

where 0 represents the sky and a column of 1s is a skyscraper. In the above example, the heights of skyscrapers are 2, 3, 4, 1, 5, 3 and 3 respectively.

Input:

First line contains 2 integers P, Q

Second line contains 2 integers M, N

Next M lines contain 1 string each of length N

Output:

First line: Total reputation of Doja Dog

Second line: Total reputation of DJ Snack

Next M lines contain a final layout of FanVille i.e. print each skyscraper owned by Doja Dog as D and skyscrapers owned by DJ Snack as S (see test case)

Example Test Case:

Input:

1 1

6 4
0000
0010
0011
0111
1111
1111

Output:
14 16

0000
00D0
00DS
0DDS
SDDS
SDDS

Explanation:
Year 1:
Doja Dog initially has 1 grammy and buys skyscraper at index 2 which has height 5 to get 5*1 = 5 reputation

Year 2:
Doja Dog and DJ Snack both win a grammy
DJ Snack now has 2 grammys and buys skyscraper at index 3 which has height 4 to get 4*2 = 8 reputation

Year 3:
Doja Dog and DJ Snack both win another grammy
Doja Dog now has 3 grammys and buys skyscraper at index 1 which has height 3 to get 3*3 = 9 reputation

Year 4:
Doja Dog and DJ Snack both win another grammy
DJ Snack now has 4 grammys and buys the last remaining skyscraper of height 2 to get 2*4 = 8 reputation

Total reputation for Doja Dog = 5+9 = 14

Total reputation for DJ Snack = 8+8 = 16

Since the skyscraper at indexes 1 and 2 are owned by Doja Dog, we turn the 1s in those skyscrapers to Ds.

Similarly, we turn 1s at indexes 0 and 3 to Ss because they are owned by DJ Snack.

0000

00D0

00DS

0DDS

SDDS

SDDS

Assumptions and clarifications:

Both Doja Dog and DJ Snack start with 0 fan reputation.

Each skyscraper must have height atleast 1.

P, Q > 0

If there are 2 skyscrapers with the same maximum height, then the singers can buy any one of them.