Mock-1 Solution

Problem-1

Accept a string of length three as input and print all possible three-letter strings that can be formed using the characters in the original string. Repetition is allowed. Print the strings in alphabetical order, one string on each line. You can assume that all characters in the input string will be unique.

Solution

```
word = ''.join(sorted(input()))
for a in word:
    for b in word:
        for c in word:
            print(a, b, c, sep = '')
```

Private Test Cases

Input	Output
abc	aaa aab aac aba abb abc aca acb acc baa bab bac bba bbb bbc bca bcb bcc caa cab cac cba cbb cbc cca ccb ccc

Input	Output
321	111 112 113 121 122 123 131 132 133 211 212 213 221 222 223 231 232 233 311 312 313 321 322 323 331 332 333

Input	Output
qpr	ppp ppq ppr pqp pqq pqr prp prq prr qpp qpq qpr qqp qqq qqr qrp qrq qrr rpp rpq rpr rqp rqq rqr rrp rrq rrr

Input	Output
109	000 001 009 010 011 019 090 091 099 100 101 109 110 111 119 190 191 199 900 901 909 910 911 919 990 991 999

Input	Output
atc	aaa aac aat aca acc act ata atc att caa cac cat cca ccc cct cta ctc ctt taa tac tat tca tcc tct tta ttc ttt

Problem-2

`numbers.txt` is a file that has a sequence of comma separated integers on each line. The first three lines of the file are given for your reference:

```
1,2,3,4,5
3,1,10,9,8,4,6
5
```

Write a function named `process_line` that accepts a non-negative integer — `i` — as input. Find the following information about line-i in the file:

- N: number of integers in the line
- S: sum of all integers in the line
- P: product of all integers in the line

Return the tuple (N, S, P).

```
def process_line(i):
    '''
        Argument: non-negative integer
        Return: (N, S, P) --- tuple of integers
    '''
```

If `i` is greater than or equal to the number of lines in the file, return the tuple `(-1, -1, -1)`. Zero-based indexing is used. So, `i = 0` corresponds to the first line in the file, `i = 1` the second line and `i = n - 1` corresponds to the last line in a file of `n` lines.

You do not have to accept input from the user or print the output to the console. You just have to write the function definition. However, within the function, you have to read the file `numbers.txt`.

Solution

```
def process_line(i):
    with open('numbers.txt', 'r') as f:
        lines = f.readlines()
        rows = [ ]
        for line in lines:
            row = [int(num) for num in line.strip().split(',')]
            rows.append(row)
        if i >= len(rows):
            return -1, -1, -1
        row = rows[i]
        S, P, C = 0, 1, len(row)
        for num in row:
            S += num
            P *= num
        return C, S, P
```

Private Test Cases

Input	Output
3	7,70,10000000
4	2,1002,1001
5	3,24,504
7	10,45,0
50	-1,-1,-1

Problem-3

`students` is a list of lists. Each element in students is a list of courses done by a student in a semester. An example list is as follows:
`[['math', 'phy', 'chem', 'cs'], ['math', 'phy'], ['math', 'chem'], ['history', 'eco']]`
The first student has done the courses `['math', 'phy', 'chem', 'cs']`, the last student has done `['history', 'eco']`. You get the idea.
Your task is to write two functions:
(1) Write a function named `consolidate` that accepts students as an argument and returns a dictionary named `consol` that has the following structure: courses are the keys; value for a given key is the number of students who have done that course.

```
def consolidate(students):
    '''
        Argument: students --- a list of lists,
                  each element of an inner list is a string (course name)
        Return: consol --- a dictionary:
                key: string (corresponds to course name)
                value: number of students who have done the above course
    '''
```

(2) Write a function named `popular` that accepts the dictionary returned by consolidate as argument. It should return the course that has been done by the maximum number of students. You can assume that there will always be exactly one such course.

```
def popular(consol):
    '''
        Argument: consol --- a dictionary:
                key: string (corresponds to course name)
                value: number of students who have done the above course
        Return: popular_course --- a string
                the name of the course which has been done by the most number
of                    students
    '''
```

You do not have to accept input from the user or print the output to the console. You just have to write the function definition for two functions: consolidate and popular. Calling the functions will be the responsibility of the autograder.

Solution

```
def consolidate(students):
    P = dict()
    for student in students:
        for course in student:
            if course not in P:
                P[course] = 0
            P[course] += 1
    return P

def popular(P):
    max_course, max_enroll = None, -1
```

```
    for course, enroll in P.items():
        if enroll > max_enroll:
            max_course, max_enroll = course, enroll
    return max_course
```

Private Test Cases

Input	Output
[['CS11', 'MA12'], ['HS23', 'MA12'], ['HS23', 'CS12'], ['HS23', 'MA11'], ['HS23', 'MA12', 'CS11', 'CS12']]	CS11:2 CS12:2 HS23:4 MA11:1 MA12:3 Popular:HS23
[['ma', 'ch', 'cs'], ['ma'], ['ch'], ['ch', 'ph'], ['ch'], ['ch', 'ma', 'cs', 'ph'], ['ma', 'ch'], ['hs']]	ch:6 cs:2 hs:1 ma:4 ph:2 Popular:ch
[['CS1', 'CS2'], ['CS3', 'CS4'], ['CS5', 'CS6', 'CS7'], ['CS1']]	CS1:2 CS2:1 CS3:1 CS4:1 CS5:1 CS6:1 CS7:1 Popular:CS1
[['CS1', 'CS2'], ['CS3', 'CS4'], ['CS5', 'CS6', 'CS7'], ['CS1'], ['CS2', 'CS3'], ['CS2'], ['CS2', 'CS4']]	CS1:2 CS2:4 CS3:2 CS4:2 CS5:1 CS6:1 CS7:1 Popular:CS2
[['math']]	math:1 Popular:math

Problem-4

This problem is about reversing a square matrix along row or column:

```
Matrix          Reverse along row       Reverse along column
1,2             3,4                     2,1
3,4             1,2                     4,3
```

The first line of the input will be an integer `n`, which denotes the dimension of the square matrix. Each of the next `n` lines in the input will have a sequence of `n` comma-separated integers. The last line in the input will be one of these two words: `row` or `column`. If it is row, then reverse the matrix along the row, else, reverse it along the column.

Print the reversed matrix as output: each line should contain one row of the matrix as a sequence of comma-separated integers.

Solution

```python
n = int(input())
mat = [ ]
for i in range(n):
    mat.append([int(word) for word in input().split(',')])
axis = input()

out = [[0 for _ in range(n)] for _ in range(n)]
if axis == 'row':
    for i in range(n):
        for j in range(n):
            out[i][j] = mat[n - i - 1][j]
else:
    for i in range(n):
        for j in range(n):
            out[i][j] = mat[i][n - j -1]


for i in range(n):
    for j in range(n):
        if j != n - 1:
            print(out[i][j], end = ',')
        else:
            print(out[i][j])
```

Private Test Cases

Input	Output
3 1,2,3 4,5,6 7,8,9 row	7,8,9 4,5,6 1,2,3
3 1,2,3 4,5,6 7,8,9 column	3,2,1 6,5,4 9,8,7
3 1,1,1 1,0,1 0,1,1 column	1,1,1 1,0,1 1,1,0
4 1,2,3,4 5,6,7,8 9,10,11,12 13,14,15,16 row	13,14,15,16 9,10,11,12 5,6,7,8 1,2,3,4
4 1,2,3,4 5,6,7,8 9,10,11,12 13,14,15,16 column	4,3,2,1 8,7,6,5 12,11,10,9 16,15,14,13

Problem-5

In spreadsheets, columns are labeled as follows:

```
Label           Number
A               1
...             ...
Z               26
AA              27
...             ...
AZ              52
BA              53
...             ...
ZZ              702
AAA             703
...             ...
AAZ             728
ABA             729

```

A is the first column, B is the second column, Z is the 26th column and so on. The three dots represent the missing labels and their column numbers. Using the table given above, deduce the mapping between column labels and their corresponding numbers. Accept the column label as input and print the corresponding column number as output.

Solution

```
mapping = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
def colnum(col):
  base = mapping.index(col[0]) + 1
  if len(col) == 1:
    return base
  return base * (26 ** (len(col) - 1)) + colnum(col[1: ])

print(colnum(input()))
```

Private Test Cases

Input	Output
ABCD	19010
AZEQT	917794
ZAERG	11902807
ABCDEFGHIJK	152686330658691
ZZZZZZZ	8353082582