Week-7, Graded, Programming

Problem-1

Question

Accept an integer as input and print the digits (as lower case words) present in it from left to right. Each digit should be printed on a separate line. At the end, print all the digits (as words) in a single line without space in camel case.

Test Cases

Type	Input	Output
public	123	one two three oneTwoThree
public	5440	five four four zero fiveFourFourZero
private	10101	one zero one zero one oneZeroOneZeroOne
private	123456789	one two three four five six seven eight nine oneTwoThreeFourFiveSixSevenEightNine
private	66666	six six six six six sixSixSixSixSix

Answer

```
num = input()
# use a dictionary to store the mapping between digits and words
D = {'0': 'zero', '1': 'one', '2': 'two', '3': 'three', '4': 'four',
     '5': 'five', '6': 'six', '7': 'seven', '8': 'eight', '9': 'nine'}
# treat num as a string
# loop through each digit and print the corresponding word
# key is digit, value is corresponding word
```

```
for digit in num:
    print(D[digit])

# go through (index, digit) pair in the number
# here, index is the position of the digit in num
# starting from 0 to len(num) - 1
# you can look up the enumerate function online
for index, digit in enumerate(num):
    if index != 0:
        # capitalize the word corresponding to the first digit
        word = D[digit].capitalize()
    else:
        # every other word remains in lower case
        word = D[digit]
    # if it is not the last word, add a comma at the end
    if index != len(num) - 1:
        print(word, end = '')
    #
    else:
        print(word)
```

Problem-2

Question

Accept details of a class of students as input from the user and create a dictionary named `students` to store the details.

Input Format

You have to accept input from the user in this problem.

- The first line of input will be a string that denotes the test case number. Store it in a string variable called `TEST_CASE`. This will be used by us for processing your solution.
- The second line of input will have the number of students in the class. Call this n.
- The third line of input will be a sequence of $s+1$ comma-separated words. The first word will always be `id`. The remaining s words will be names of the subjects taken up by the class.
- The next n lines will have $s+1$ comma-separated numbers on each line. The first number will be the student id. The remaining s numbers on this line will have the marks scored by this student in the corresponding subjects.

Sample Input

```
TEST CASE 1
2
id,sub1,sub2,sub3
1,85,65,90
2,100,56,34
```

Evaluation Format

You are not expected to print anything to the console. You only have to create a dictionary named `students` that is structured as follows.

- Keys will be student ids. It has to be an integer.
- Value for each key will be another dictionary which is structured as follows:
  - Keys will be subject names and should be strings.
  - Values will be the marks scored by the student in that subject. All values will be integers.

Sample Output

`students` is the dictionary's name:

```
{
    1: {
        'sub1': 85,
        'sub2': 65,
        'sub3': 90
        }
    2: {
        'sub1': 100,
        'sub2': 56,
        'sub3': 34
       }
}
```

Note

- You have to accept input from the user in this problem.
- You do not have to print the output to the console.
- You will be tested based on the dictionary `students` that you create. Do NOT use a different name for the dictionary. If `students` matches the exact format given in the problem specification, you will get the actual output as `CORRECT`.
- Do NOT forget to store the test case string in the variable `TEST_CASE`.
- `TEST_CASE` and `students` should have global scope so that we can call them from anywhere from within the code.

Test Cases

Type	Input	Output
Public	TEST CASE 1 2 id,sub1,sub2,sub3 1,85,65,90 2,100,56,34	CORRECT
Public	TEST CASE 2 5 id,maths,physics,chemistry,biology,compsci 1,10,20,30,40,50 2,60,70,80,90,100 3,100,98,38,49,29 4,100,100,100,100,100 5,78,39,98,29,98	CORRECT
Private	TEST CASE 3 1 id,s1,s2 1,78,45	CORRECT
Private	TEST CASE 4 10 id,Maths 1,10 2,20 3,30 4,40 5,50 6,60 7,70 8,80 9,90 10,100	CORRECT

Answer

```
# accept test case
TEST_CASE = input()
# accept the number of students as input
n = int(input())
# names of subjects to be stored as a list
subjects = [ ]
# get the subject names
for word in input().split(','):
    if word == 'id':
        continue    # ignore the id word
```

```
    subjects.append(word)
# student details to be stored in students dict
students = dict()
for i in range(n):
    # input sequence
    inp_seq = input().split(',')
    # first entry is student id
    sid = int(inp_seq[0])
    # create a key for this student
    students[sid] = dict()
    # go through all marks
    for i in range(len(inp_seq[1:])):
        # this is the subject name
        subject = subjects[i]
        # get the mark for this subject
        mark = int(inp_seq[i + 1])
        # for student sid, and for subject, add the value mark
        students[sid][subject] = mark
```

Problem-3

Question

Write a function named `merge` that accepts two dictionaries as input, merges them into a single dictionary, and returns merged dictionary as output. This is what we mean by merging:

- The key-value pairs in both dictionaries should be present in the returned dictionary.
- If a particular key is common to both dictionaries, the value corresponding to this key in one of the two dictionaries should be retained.

Input Format

- The only input for this problem corresponds to the test case number. You do not have to accept this from the user. We will take care of this.
- The input dictionaries will not be shown to you. They will be passed as arguments to your functions. We will take care of this.

Evaluation Format

You have to write the function `merge`

- The function `merge` accepts three arguments.
  - `D1`: first dictionary
  - `D2`: second dictionary
  - `priority`: The is a string variable that denotes the priority given to common keys while merging. That is, if both `D1` and `D2` have a key in common, then this variable will determine which value needs to be retained. More specifically, `priority` can take one of these two values:
    - `first`: retain the value corresponding to the common key present in the first dictionary
    - `second`: retain the value corresponding to the common key present in the second dictionary

```
def merge(D1, D2, priority):
    '''
        Arguments:
            - D1: first dictionary
            - D2: second dictionary
            - priority: string
        Returns: D; merged dictionary
    '''
```

- You do not need to print output to the console.
- The dictionary returned by your function will be evaluated against our solution. If it matches, then the expected output is `CORRECT`.

Note

- You do NOT have to call the function `merge`. The function call will be our responsibility.

Answer

```python
def merge(D1, D2, priority):
    # basic idea is to write code only for priority being first
    # when priority is second, we can just have a recursive call
    # with priority being first
    if priority == 'second':
        return merge(D2, D1, 'first')
    # merged dict will be stored in D
    D = dict()
    # first add all key-value pairs in D1 into D
    for key, value in D1.items():
        D[key] = value
    for key, value in D2.items():
        # if this is not a common key, only then add it to D
        if key not in D:
            D[key] = value
    return D
```

Test Cases

```python
data = dict()
data['TEST CASE 1'] = {
    'D1': {1: 2, 2: 3, 3: 4},
    'D2': {1: 10, 4: 15, 5:10},
    'priority': 'first'
}
data['TEST CASE 2'] = {
    'D1': {1: 2, 2: 3, 3: 4},
    'D2': {1: 10, 4: 15, 5:10},
    'priority': 'second'
}
data['TEST CASE 3'] = {
    'D1': {'a': 1, 'b': 100, 'c': 1000},
    'D2': {'d': 20, 'e': 1000, 'c': 1234},
    'priority': 'first'
}
data['TEST CASE 4'] = {
    'D1': {'a': 1, 'b': 100, 'c': 1000},
    'D2': {'d': 20, 'e': 1000, 'c': 1234},
    'priority': 'second'
}
data['TEST CASE 5'] = {
    'D1': {(0, 1): [1, 2, 3], (1, 2): [4, 5, 6], (3, 4): [1000]},
    'D2': {(0, 1): [5, 6, 7], (1, 2): [10, 11, 12], (5, 6): [1234]},
    'priority': 'first'
}
data['TEST CASE 6'] = {
    'D1': {(0, 1): [1, 2, 3], (1, 2): [4, 5, 6], (3, 4): [1000]},
    'D2': {(0, 1): [5, 6, 7], (1, 2): [10, 11, 12], (5, 6): [1234]},
    'priority': 'second'
}
```

Problem 4

Question

Suppose you have an $m \times n$ matrix, where m represents number of rows and and n represents number of columns and $2 < m, n < 12$, that consists of integer values. Write a program that creates a new $m \times n$ matrix in which all the elements are replaced by zeros except the border elements.

Input Format

- The first line contains a positive integer `m` to represent the number of rows.
- The second line contains a positive integer `n` represent to number of columns.
- Accept `m` rows in which each row contain `n` elements separated by space.

Evaluation Format

Print the matrix elements row wise and separated by space after replacing all elements by zeros except the border elements.

Sample Input

```
4
4
1 2 3 4
2 3 4 5
3 4 5 6
4 5 6 7
```

Sample Output

```
1 2 3 4
2 0 0 5
3 0 0 6
4 5 6 7
```

Answer

```
# This function accepts data row-wise from the user and returns two-dimension matrix as a list
def create_matrix(r,c):
    l1 = []
    i = 0;
    while i < r:
        l2 = input().split(' ')
        l1.append(l2)
        i += 1
    return l1
# This function replaces all elements by zero except the border elements
def replacebyzero(l,r,c):
    for i in range(r):
        for j in range(c):
            # Below conditional expression target to non-border elements of the matrix
            if i!=0 and j!=0 and i!=r-1 and j!=c-1:
```

```
                l[i][j]=0;

# Accept the number of row and column from user
m = int(input())
n = int(input())
# Call create_matrix() function to accept data row-wise from user and return two-dimension matrix as a list
l = create_matrix(m,n)
# Call replacebyzero() function that replace all element by zero except the border elements
replacebyzero(l,m,n)

# Print the matrix elements row-wise and separated them by space
for i in range(m):
    for j in range(n):
        if j != n-1:
            print(l[i][j],end = ' ')
        else:
            print(l[i][j])
```

Test Cases

Public

Input-1

```
4
4
1 2 3 4
2 3 4 5
3 4 5 6
4 5 6 7
```

Output-1

```
1 2 3 4
2 0 0 5
3 0 0 6
4 5 6 7
```

Input-2

```
4
3
2 3 4
2 3 4
4 6 7
8 7 5
```

Output-2

```
2 3 4
2 0 4
4 0 7
8 7 5
```

Private

Input-1

```
5
5
1 2 3 4 5
-1 -2 -3 -4 -5
7 8 9 4 5
2 1 3 4 5
1 4 7 8 9
```

Output-1

```
1 2 3 4 5
-1 0 0 0 -5
7 0 0 0 5
2 0 0 0 5
1 4 7 8 9
```

Input-2

```
6
4
0 0 0 0
0 1 2 0
0 2 5 0
0 1 5 0
0 2 5 0
0 0 0 0
```

Output-2

```
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
```

Input-3

```
4
4
1 2 3 4
3 0 0 5
4 0 0 6
2 3 4 5
```

Output-3

```
1 2 3 4
3 0 0 5
4 0 0 6
2 3 4 5
```

Problem 5

Question

The `Collatz` function is defined for a positive integer n as follows.

- $f(n) = 3n + 1$ if n is odd
- $f(n) = \frac{n}{2}$ if n is even

We consider the repeated application of the `Collatz` function starting with a given integer n, as follows:

$f(n), f(f(n)), f(f(f(n))), \cdots$

It is conjectured that no matter which positive integer n you start from, the sequence will always reach 1. [Source: Wikipedia]

For example, If $n = 10$, the sequence is:

Seq No.	x	f(x)
1	10	5
2	5	16
3	16	8
4	8	4
5	4	2
6	2	1

Thus if you start from $n = 10$, you need to apply the function f six times in order to first reach 1.

Write a function `collatz_repeat(n)`, where `1 < n <= 32,000`, and return the number of times f has to be applied repeatedly in order to first reach 1.

Input Format

- The input to the program will be an integer in the range $1 < n \leq 32000$. You do NOT have to accept input from the user.

Evaluation Format

Complete the following function:

```
def collatz_repeat(n):
    '''
        Argument: n -> integer
        Returns: integer
    '''
```

- You do NOT have to print the output to the console. This will be done by us internally.
- You do not have to call the function. It is sufficient if you complete the body of the function.

Answer

Recursive Solution

```
def collatz_repeat(n):
    # When n becomes 1, 0 will be added in the return value and this is the terminate    condition for collatz_repeat(n)
    if n == 1:
        return 0
    else:
        # For each recursive call of collatz_repeat(n), 1  will be added in return          value
        if n % 2==0:
            return 1 + collatz_repeat(n//2)
        else:
            return 1 + collatz_repeat(3*n+1)
```

Non-Recursive Solution

```
def collatz_repeat(n):
    # Initialize counter count to 0
    count=0
    while n != 1:
        if n % 2 == 0:
            # if n is even then n//2 will be assign to n
            n = n // 2
            # Increase the count value by 1
            count += 1
        else:
            # if n is odd then 3 * n + 1 will be assign to n
            n = 3 * n + 1
            # Increase the count value by 1
            count += 1
    # Return count after completion of the while loop
    return count
```

Suffix Code Block (hidden)

```
n = int(input())
print(collatz_repeat(n))
```

Test Cases

Public Test Cases

Input	Output
`101`	`25`
`100`	`25`
`2463`	`208`
`7`	`16`

Private Test Cases

Input	Output
`30000`	`178`
`31999`	`98`
`125`	`108`
`999`	`49`