Question 1 - Lists

6. (8 points) Hot Cat (Like a hot dog, but with little cat ears on the end.)

Implement `compress`, which takes a deep list of integers and returns a *new list* compressing all neighboring integers in the input list. Compression involves reducing a group of neighboring integers to a single number whose value is the sum of the group. Integers in a list are considered neighbors if their indices differ by 1.

Compressing `[1, 2, 3]` results in `[6]` since the input integers are all part of a group of neighboring integers.

```
def compress(lst):
    """Given a deep list of integers, return a new list compressing all neighboring integers.

    >>> compress([])
    []
    >>> compress([1, 2, 3])
    [6]
    >>> compress([0, 0, 0, 0])
    [0]
    >>> compress([1, 2, [3, 4]])
    [3, [7]]
    >>> compress([[11, 12], 3, 4, [1, 2], [5, 6], 7, 8, [9, 10]])
    [[23], 7, [3], [11], 15, [19]]
    >>> compress([1, 2, [3, [4, 5, 6], [7, 8], 9, 10], 11, 12])
    [3, [3, [15], [15], 19], 23]
    """
```

(Attempt only after previous question is finished)

8. (10 points) Annoying Dog (A little white dog. It's fast asleep. . .)

(a) (2 pt) Implement a `list_counter` that returns a number in base 10 equal to the value of the `digits` in the given `base`. Numbers that are not digits in the given base are ignored. Each subsequent digit increases the value of the preceding digits by a factor of `base`.

The value of `list_counter(2, [1, 0, 1, 1])` is computed by reading the `digits` from left to right:

$$\left[\left(\left[\left(\mathbf{1}\right)\cdot 2\right]+\mathbf{0}\right)\cdot 2\right]+\mathbf{1}\Big)\cdot 2\Big]+\mathbf{1}$$

```
def list_counter(base, digits):
    """Return a number in base 10 equal to the value of the digits in the given base.
    Numbers that are not digits in the given base are ignored.

    >>> list_counter(2, [])
    0
    >>> list_counter(2, [1, 0, 1, 1])     # see example above
    11
    >>> list_counter(2, [1, 2, 3, 0, 1])  # 2 and 3 are not digits in base 2
    5
    >>> list_counter(4, [1, 2, 3, 0, 1])  # 1*(4**4) + 2*(4**3) + 3*(4**2) + 0*(4**1) + 1*1
    433
    """
```

Question 2 - Loops

An integer d is a *divisor* of an integer n if the remainder of $n \div d = 0$.

Given an integer, for each digit that makes up the integer determine whether it is a divisor. Count the number of divisors occurring within the integer.

Note: Each digit is considered to be unique, so each occurrence of the same digit should be counted (e.g. for $n = 111$, 1 is a divisor of 111 each time it occurs so the answer is 3).

Input Format

The first line is an integer, t, indicating the number of test cases.
The t subsequent lines each contain an integer, n.

Constraints

$1 \leq t \leq 15$
$0 < n < 10^9$

Output Format

For every test case, count the number of digits in n that are divisors of n. Print each answer on a new line.

Sample Input

```
2
12
1012
```

Sample Output

```
2
3
```

Question 3 - Dictionary and Files

Write a program that takes in a file `user-info.txt` that contains the following information - (copy-paste this into a file on your computer)

```
A password1 B Y E
B password2 P O B N A C
C password3 B O Y
D password4 F P U E
E password5 A D Y N
Y password6 A C E N
U password7 D P
N password8 F B E Y
F password9 D N
O password10 B C
P password11 D B U
```

The first letter/alphabet is a userID, followed by user-password and followed by a list of friends.

Example - 'A' is a user, A's password is 'password1' and A's friends are 'B', 'Y', 'E'.

For this program write the following functions -

a. def mutual_friends(user1, user2) - this function takes in two user ids of separate users and outputs the mutual friends.
b. def login() - this prompts the users to enter userID and password. If the input given is incorrect, provide a suitable error message. (Bonus: use String Formatting)
c. def sign_up() - prompt user to sign up by submitting userID and password. Append user info to the file and DO NOT allow existing users to sign up.
d. def add_friends(user1, user2) - makes user1 and user2 friends.
e. def re_write() - this method rewrites information in the file to show added friends

Question 4

Values of different coins are = 1-cent, 2-cent, 4-cent, 8-cent …..

Find the number of ways you can create change for a given amount.

For example:-

If `amount = 7`

Then, the number of ways are -

1. 7*1-cent coin
2. 5*1-cent coin, 1*2-cent coin
3. 3*1-cent coin, 2*2-cent coin
4. 3*1-cent coin, 1*4-cent coin
5. 1*1-cent coin, 3*2-cent coin
6. 1*1-cent coin, 1*2-cent coin, 1*4-cent coin

```
# Code Skeleton -

def count_change(amount):

     return None

print(count_change(3)) # expected answer is 2
print(count_change(7)) # expected answer is 6
print(count_change(10)) # expected answer is 14
print(count_change(20)) # expected answer is 60
```