1

2

3

4

5

6

☆ Keypad

Jesse is trying out different variations of the old keypad similar to the one in the image below. There are 3 or 4 letters assigned to each key. To enter the first letter of any group, that key is pressed once. For the second letter, the key is pressed twice and so on for the third and fourth letters.

1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
*
0 +
#

He arranges the letters and numbers such that the letters are present only on the numbers from 2-9. Two of the numbers have exactly 4 characters and the rest of them have 3 characters. The number 1 always represents the space character, ascii(32). Given a string *message*, that consists of uppercase characters and spaces, Jesse wants to print the numeric version of it. Determine how many different messages that contain characters and spaces can be formed from the generated sequence of keypresses, modulo *1000000007*.

For example, the string to input, *message = 'ME'*. Key values are as follows:

```
1 [space]
2 Q Z P
3 H D F X
4 J N M
5 T E O
6 B S U
7 I W G
8 V C K Y
9 L A R
```

The 'M' is entered by pressing the '4' key *3* times. The 'E' is entered by pressing the '5' key *2* times. The string of keypresses is *'44455'*. Pressing the '4' key *3* times could be used to enter 'JJJ', 'JN', 'NJ' or 'M'. Pressing the '5' key twice can represent 'TT' or 'E'. There are *8* strings that can be entered as '44455':

```
JJJTT   JJJE
JNTT    JNE
NJTT    NJE
MTT     ME
```

***Note:** The key/character mapping above does not match the input format. Key '1' is shown and key numbers were added for clarity.

Function Description

countMessages has the following parameters:

keys: an array of *8* strings

message: a string to generate keypresses for

1
2
3
4
5
6

Constraints

- $|keys| = 8$
- $|keys[i]| \in [3, 4]$
- $1 \le |message| \le 10^5$
- $|message[i]| \in [A\text{-}Z, ' ']$

Input Format for Custom Testing

Sample Case 0

Sample Input 0

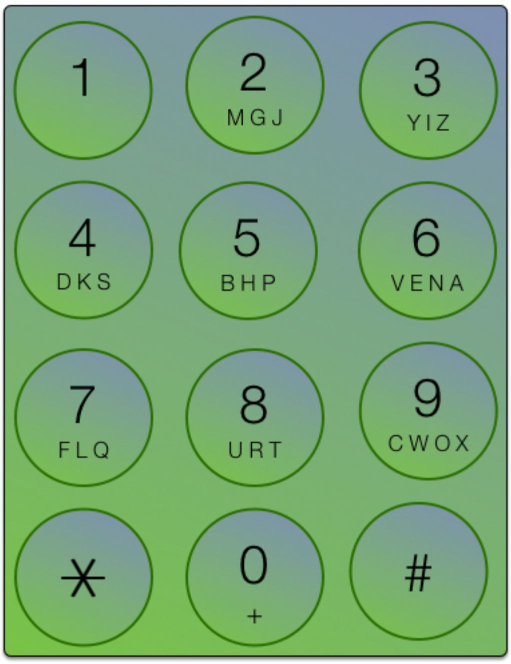
```
8
MGJ
YIZ
DKS
BHP
VENA
FLQ
URT
CWOX
HEY
```

Sample Output 0

```
4
```

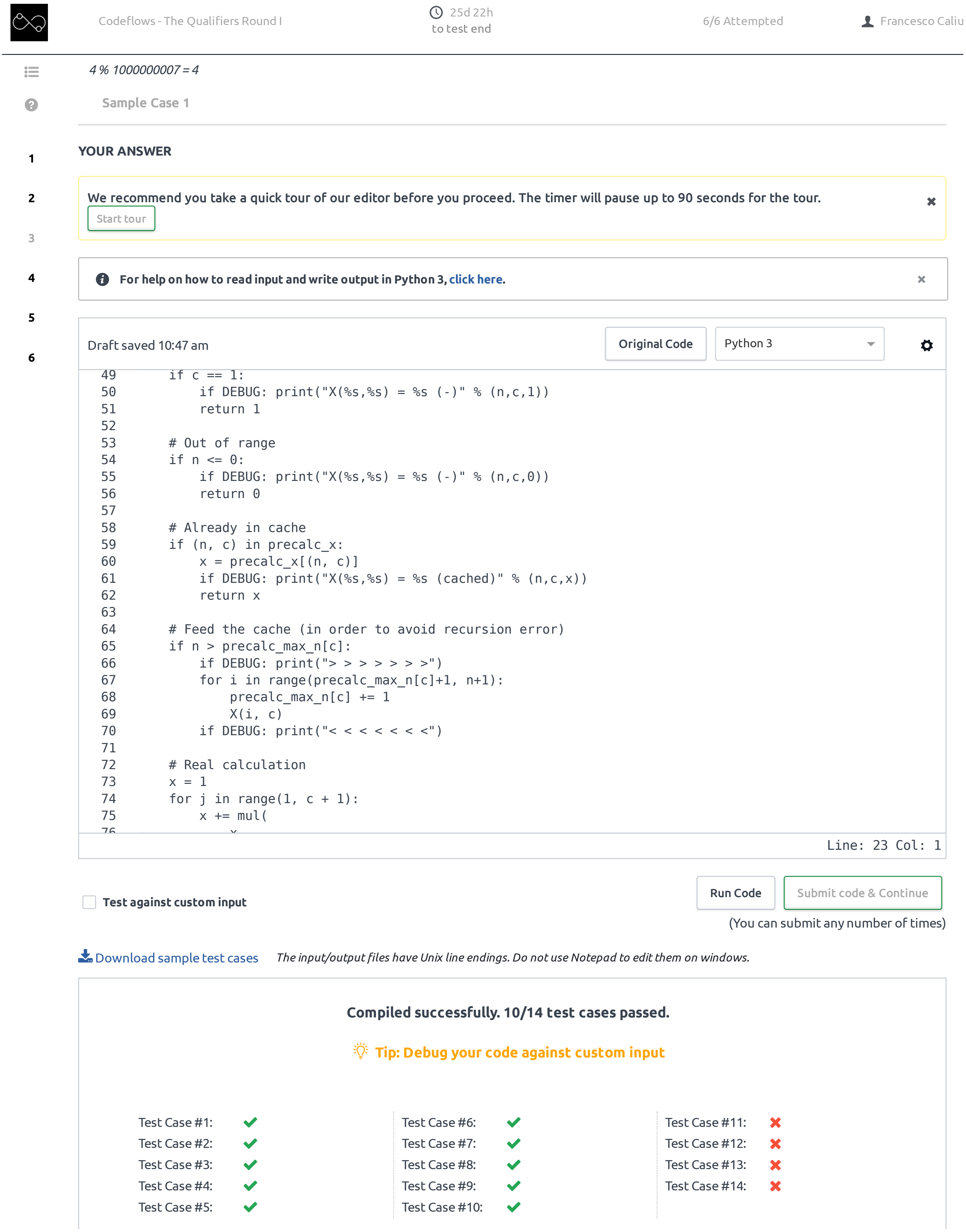
Explanation 0

The keypad looks like this:



The numeric version of the message `HEY` is `55663` The four messages that can be formed from the numeric message `55663` are:

```
BBVVY
BBEY
```



4 % 1000000007 = 4

Sample Case 1

YOUR ANSWER

We recommend you take a quick tour of our editor before you proceed. The timer will pause up to 90 seconds for the tour.

For help on how to read input and write output in Python 3, click here.

Draft saved 10:47 am

```
    if c == 1:
        if DEBUG: print("X(%s,%s) = %s (-)" % (n,c,1))
        return 1

    # Out of range
    if n <= 0:
        if DEBUG: print("X(%s,%s) = %s (-)" % (n,c,0))
        return 0

    # Already in cache
    if (n, c) in precalc_x:
        x = precalc_x[(n, c)]
        if DEBUG: print("X(%s,%s) = %s (cached)" % (n,c,x))
        return x

    # Feed the cache (in order to avoid recursion error)
    if n > precalc_max_n[c]:
        if DEBUG: print("> > > > > > >")
        for i in range(precalc_max_n[c]+1, n+1):
            precalc_max_n[c] += 1
            X(i, c)
        if DEBUG: print("< < < < < < <")

    # Real calculation
    x = 1
    for j in range(1, c + 1):
        x += mul(
```

Line: 23 Col: 1

□ Test against custom input

(You can submit any number of times)

Download sample test cases *The input/output files have Unix line endings. Do not use Notepad to edit them on windows.*

Compiled successfully. 10/14 test cases passed.

Tip: Debug your code against custom input

Test Case #1:	✓	Test Case #6:	✓	Test Case #11:	✗
Test Case #2:	✓	Test Case #7:	✓	Test Case #12:	✗
Test Case #3:	✓	Test Case #8:	✓	Test Case #13:	✗
Test Case #4:	✓	Test Case #9:	✓	Test Case #14:	✗
Test Case #5:	✓	Test Case #10:	✓		

1

2

3

4

5

6

Input [**Download**]

```
8
MGJ
YIZ
DKS
BHP
VENA
FLQ
URT
CWOX
HEY
```

Your Output

```
4
```

Expected Output [**Download**]

```
4
```

Testcase 2: *Success*

Input [**Download**]

```
8
LAP
RHO
IVQW
SKJ
TZU
MDX
NGYC
BEF
LEARN IT
```

Your Output

```
4
```

Expected Output [**Download**]

```
4
```

Testcase 3: *Success*

Input [**Download**]

```
8
WPL
HTVG
QIN
YSMX
KAF
OCJ
UZR
DEB
OCWQQDH
```

Your Output

```
8
```

Expected Output [**Download**]

```
8
```

Testcase 4: *Success*

Your Output

```
Output hidden
```

1

2

3

4

5

6

```
Output hidden
```

Testcase 6: *Success*

Your Output

```
Output hidden
```

Testcase 7: *Success*

Your Output

```
Output hidden
```

Testcase 8: *Success*

Your Output

```
Output hidden
```

Testcase 9: *Success*

Your Output

```
Output hidden
```

Testcase 10: *Success*

Your Output

```
Output hidden
```

Testcase 11: *Wrong Answer*

Your Output

```
Output hidden
```

Testcase 12: *Wrong Answer*

Your Output

```
Output hidden
```

Testcase 13: *Wrong Answer*

Your Output

```
Output hidden
```

Testcase 14: *Wrong Answer*

Your Output

```
Output hidden
```