Timed Lab 1 – Flip Phone

This is a Timed Lab; this Timed Lab is worth 22 **Exam** points.

The TAs will be available to answer clarifying questions about the problem, but they are not permitted to give assistance with debugging the code, program logic, etc. You will have an entire recitation period to work on this assignment; this time begins *exactly* when your recitation begins and ends *exactly* when your recitation ends. No extra time will be given if you arrive late, except in highly extenuating circumstances that must be approved by Dr. Summet.

T-Square will not permit any late submissions; ensure that you submit your code to T-Square several times to prevent earning a zero due to you being unable to submit. Your TAs will give a verbal warning 10 and 5 minutes before the end of the recitation period; you should submit at these times. In addition, before leaving your recitation section, be sure to have received the confirmation email of your T-Square submission!

In your collaboration statement, if you use code from somewhere that is **not** a class resource (i.e. not listed on the course calendar), please list where this code came from. Ensure that you fill out the header at the top of the file.

Note that you must check out with your TA before you leave the recitation room. If you do not check out with your TA or you modify your submission after you leave the recitation room, **you will receive a grade of zero on the timed lab**. No submissions will be accepted after T-Square closes the assignment (i.e. it will not let you submit).

Problem Description:

When using older cell phones, one must press several keys to record even a single alpha-numeric character, this is called the multi-tap system. This led to complicated processes to assemble one piece of text and to many users wearing out their phone keypads! You've been assigned to translate a simple text message into the process of key presses and to figure out a way to tell a user which key(s) he or she may be wearing out.

Function keyPress:

The first function you need to write takes in a piece of alpha-numeric text as a parameter and will return the string of multi-tap keystrokes needed to type that text on a keypad. You can assume that all text is made up of lowercase letters a-z, numbers 0-9, and spaces and nothing else.

For keys with letters on them, multiple key presses are needed to access letters after the first and for numbers. Here are some examples:

- 'p' -> '7'
- 'q' -> '77'
- 'r' -> '777'
- 's' -> '7777'
- '7' -> '77777'

Some keys like '1', '0', '*', and '#' do not represent alpha characters and instead represent:

- '1' -> '1'
- '0' -> '0'
- ' ' (space) -> '*'
- Separator -> '#'

Anytime two characters are needed in a row that require the same key to be pressed, a '#' must be used to separate them. See example 'no' below for clarification on use of the separator. Remember to return your multi-tap keystroke string! *It might be useful, but not required, to hardcode these translations into a dictionary. Remember your multi-tap string will have no spaces or letters.*

Test Cases:

```
>>> keyPress('no')
'66#666'
>>> keyPress('a p p')
'2*7*7'
>>> keyPress('hello world')
'4433555#555666*96667775553'
>>> keyPress('cs 2316')
'2227777*2222333316666'
```

Function wornOutKey:

This second function takes a multi-tab string parameter such as the output of keyPress that describes a sequence of digit key entries on a keypad. Your job is to find out which keys were used the most and how many times. If there are multiple keys used the same most amount of times, they will all be printed in the final statement back to the user. Your function will print the statement "You've worn out the key(s) <key or list of keys here> after being used <number of key presses here> times!" The type of key or keys can be displayed in any way so long as they are all present. In the examples, they are displayed as part of a list.

Test Cases:

```
>>> wornOutKey('666#66')
You've worn out the key(s) ['6'] after being used 5 times!
>>> wornOutKey('2*7*7')
You've worn out the key(s) ['7', '*'] after being used 2
times!
>>> wornOutKey('4433555#555666*96667775553')
You've worn out the key(s) ['5'] after being used 9 times!
>>> wornOutKey('2227777*2222333316666')
You've worn out the key(s) ['2'] after being used 7 times!
```

Grading:

You will earn points as follows for each piece of functionality that works correctly according to the specifications.

keyPress		**13**
Correct function header	2	
Converts text to keystrokes	5	
Inserts '#' to separate sequential key characters	3	
Returns string	1	
Returns correct multi-tap keystrokes string	2	
wornOutKey		**9**
Correct function header	1	
Counts strokes of all keys (0-9, *, and #)	3	
Prints final statement to user	1	
Correct most used key(s)	2	
Correct number of key presses on most used key(s)	2	