Quiz 6

To get credit for this quiz, use the Quiz tool at eee.uci.edu to enter your answers, within the Sunday-to-Tuesday quiz period.

Problem 1 (4 points)

(a) (4 points) A quiz has scores in the range 0 to 10. We can represent the distribution of scores on this quiz as a list of numbers, each number being the count of students who received a particular score. So in the list below, 1 person scored 0, 3 people scored 5, and 45 people scored 10:

```
counts = [1, 0, 0, 2, 2, 3, 8, 22, 33, 40, 45]
```

Suppose we want to print these statistics in a table in the following format:

```
0.   1 ( 0.64%)
1.   0 ( 0.00%)
2.   0 ( 0.00%)
3.   2 ( 1.28%)
4.   2 ( 1.28%)
5.   3 ( 1.92%)
6.   8 ( 5.13%)
7.  22 (14.10%)
8.  33 (21.15%)
9.  40 (25.64%)
10.  45 (28.85%)
```

In the following code, fill in each blank with one character so that the output is formatted as shown above.

```python
topscore = 10
for s in range(topscore + 1):
    print("{:_____d}. {:3d} ({:_____ . _____  _____}%)".format(s, counts[s], counts[s]/sum(counts)*100))
```

(b) (4 points) Suppose we want to print a simple bar graph with the table of statistics:

```
0.   1 ( 0.64%) *
1.   0 ( 0.00%)
2.   0 ( 0.00%)
3.   2 ( 1.28%) **
4.   2 ( 1.28%) **
5.   3 ( 1.92%) ***
6.   8 ( 5.13%) ********
7.  22 (14.10%) ********************
8.  33 (21.15%) ****************************************
9.  40 (25.64%) ****************************************
10.  45 (28.85%) ****************************************
```

Rewrite the code above to produce the bar graph as shown.
Problem 2 (6 points)
Complete the definition of \texttt{seconds\_to\_mmss} below, consistent with its header, docstring, and assertions. [Note: The integer division operator \((a/b)\) gives the integer quotient of \(a/b\). The mod operator \(\%\) gives the remainder of \(a/b\).] You do not have to worry about leading zeroes (like "11:05").

\begin{verbatim}
def seconds_to_mmss(seconds: int) -> str:
    ''' Convert a number of seconds to minutes and seconds in "mm:ss" format '''
    return str(seconds//60) + ':' + str(seconds % 60)
    # Alt: return '{:d}:{:2d}'.format(seconds//60, seconds % 60)
    # Alternative that fixes leading zeroes without \texttt{zfill()}: return '{:d}:{:02d}'.format(seconds//60, seconds % 60)

assert(seconds_to_mmss(15) == "0:15")
assert(seconds_to_mmss(75) == "1:15")
assert(seconds_to_mmss(3620) == "60:20")
\end{verbatim}

Problem 3 (10 points)
Parts of this excerpt from \texttt{help(str)} may be useful in this problem:

\begin{verbatim}
find(...)  
    S.find(sub) -> int
    Return the lowest index in \(S\) where the string \(sub\) is found.
    Return -1 on failure.

split(...)  
    S.split(sep) -> list of strings
    Return a list of the words in \(S\), using \(sep\) as the delimiter string.

strip(...)  
    S.strip() -> str
    Return a copy of the string \(S\) with leading and trailing whitespace removed.
\end{verbatim}

Complete the function definition below, consistent with its header, docstring, and assertions.

\begin{verbatim}
MONTHS = ['January', 'February', 'March', 'April', 'May', 'June',
          'July', 'August', 'September', 'October', 'November', 'December']

def mmddyy_to_MonthDayYear(mmddyy: str) -> str:
    ''' From an argument in the form '10/31/15' (month, day, year),
    return a string in the form 'October 31, 2015'. Assume all
    values are valid numbers and all years are in this century
    (that means your function doesn't have to check).
    '''

assert(mmddyy_to_MonthDayYear('10/31/15') == 'October 31, 2015')
assert(mmddyy_to_MonthDayYear('12/1/07') == 'December 1, 2007')
assert(mmddyy_to_MonthDayYear('1/3/99') == 'January 3, 2099')
\end{verbatim}
Problem 4 (10 points)

Suppose we wish to process text files that contain some "front matter"—lines at the start of the file that we wish to ignore, similarly to a part of this week's lab. Let's say that we have read the file into a list of strings, that the end of the front matter is indicated by a line in the file that says "END OF FRONT MATTER", and that we are guaranteed that this line will occur in the file.

Complete the definition of remove_front_matter below, consistent with its header, docstring, and assertions. [Recall that the annotation [str] means the same things as 'list of str'. Note that no actual file-handling commands are required for this solution.]

def remove_front_matter(linelist: [str]) -> [str]:
    ''' Return input list with starting lines (through "END OF FRONT MATTER") removed '''
    result = []
    dividing_line = 0
    for line in linelist:
        if line == "END OF FRONT MATTER":
            dividing_line += 1
            break
    for line in linelist[dividing_line + 1:]
        result.append(line)
    return result

test_list = ["To be skipped",
            "Also to be skipped",
            "END OF FRONT MATTER",
            "To be included",
            "Also to be included"]
assert(remove_front_matter(test_list) == ["To be included",
                                         "Also to be included"])  
assert(remove_front_matter(test_list[2:]) == ["To be included",
                                            "Also to be included"])  
assert(remove_front_matter(test_list[:3]) == [ ])


Problem 5  (11 points)
Suppose we have a list of scores on a quiz, one score for each student, in the range 0 to 20. For example:

\[
\text{quiz\_scores} = [18, 20, 18, 20, 0, 10, 10, 20, 10, 20]
\]

We would like to produce a list of counts, one count for each possible score

\[
\text{quiz\_counts} = [1, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 2, 0, 4]
\]

(a)  (4 points) Write the function \textit{zero\_counts} that takes a number (such as the number of points on a quiz) and returns a list of zeros, one zero for each possible score).

\[
\text{def zero\_counts (top\_value: int) \rightarrow 'list of int':}
\]

\[
\text{ ''' Return a list of zeroes, with one zero for each possible score from zero to}
\]

\[
\text{ top\_value}
\]

\[
\text{'''}
\]

\[
\text{assert zero\_counts(10) == [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 4]}
\]

\[
\text{assert zero\_counts(0) == [0]}
\]

(b)  (3 points) In one sentence, why does \texttt{zero\_counts(10)} return a list of \textit{eleven} zeroes?

(c)  (4 points) Now, write the function \textit{count\_scores} that takes a list of scores and a number that rep-\texttt{resents the highest possible score; it returns a list of counts, indicating how many times each score oc-\texttt{curred:}}

\[
\text{def count\_scores(scores: 'list of int', top\_score: int) \rightarrow 'list of int':}
\]

\[
\text{ ''' Return a list that tallies the number of times each value (from 0}
\]

\[
\text{ to top\_score) occurs in the list of scores}
\]

\[
\text{'''}
\]

\[
\text{assert count\_scores([], 5) == [0, 0, 0, 0, 0, 0]}
\]

\[
\text{assert count\_scores(quiz\_scores, 20) == quiz\_counts}
\]