Question 1.  [5 marks]

Beside each code fragment in the table below, write what is printed. If the code would cause an error, write ERROR and give a brief explanation.

tweet1 = '#uoft_cs Turing award winner Steve Cook'
tweet2 = 'Want cheap snacks? Visit @cssu office in BA2283'

<table>
<thead>
<tr>
<th>Code</th>
<th>Output or Cause of Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>print('Turing' in tweet1)</td>
<td>True</td>
</tr>
<tr>
<td>print('in' in tweet1 and 'in' in tweet2)</td>
<td>True</td>
</tr>
<tr>
<td>print(tweet1[1] + tweet2[-3])</td>
<td>u2</td>
</tr>
<tr>
<td>print(tweet1[tweet2[-1]])</td>
<td>ERROR: string indices must be integers</td>
</tr>
<tr>
<td>print(tweet2[tweet2.find('@'):])</td>
<td>@cssu office in BA2283</td>
</tr>
</tbody>
</table>

Question 2.  [2 marks]

Complete the example function calls by adding arguments that result in the return values shown. (For each example call, there are several correct answers, and providing any one of them will earn full marks.)

```python
def mystery(L):
    ''' (list of str) -> str

    # a list containing one string that starts and ends with 'z'
    >>> mystery(['z'])
    'zz'
    
    # a list containing three strings that start and end with
    # 'a' and 'c', 'd' and 'f', 'g' and 'i', respectively.
    >>> mystery(['abc', 'def', 'ghi'])
    'acdfgi'
    
    result = ''
    for item in L:
        result = result + item[0] + item[-1]

    return result
```
Question 3.  [4 marks]

Read the function header and body and then complete the docstring. Give a meaningful function name, the type contract, the description, and two examples that return different values.

def get_first_digit(s):
    """ (str) -> str
    Return the first digit in s, or the empty string if there are no digits in s.
    >>> get_first_digit('abc')
    '"
    >>> get_first_digit('1a2b3c4d')
    '1'
    """
    i = len(s) - 1
    x = ''
    while i >= 0:
        if s[i].isdigit():
            x = s[i]
            i = i - 1
    return x
Question 4.  [5 marks]

Some programming languages use the camel case naming style for variable and function names. In this style, words in a name after the first word start with an uppercase letter and all other letters are lowercase (e.g., camelCase). Python, on the other hand, uses the pothole case naming style, where all letters are lowercase, with an underscore between each word and digit (e.g., pothole_case). Complete this function according to its docstring description.

```python
def camel_to_pothole(camel):
    """ (str) -> str
    Precondition: len(camel) >= 1 and camel[0].isalpha() and camel.isalnum()
    Return a new string with camel converted to the pothole case naming style.
    >>> camel_to_pothole('computerScience')
    'computer_science'
    >>> camel_to_pothole('numVowels30')
    'num_vowels_3_0'
    """

    pothole = ''
    for ch in camel:
        if ch.isupper():
            pothole = pothole + '_' + ch.lower()
        elif ch.isdigit():
            pothole = pothole + '_' + ch
        else:
            pothole = pothole + ch

    return pothole
```
Question 5. [4 marks]

A mini-tweet is a string that is 1 to 10 characters long. Using the constant MAX_LENGTH when possible, complete the function according to its docstring description.

MAX_LENGTH = 10

```python
def split_into_minitweets(s):
    """ (str) -> list of str
    
    Precondition: len(s) >= 1
    
    Return a list containing the mini-tweets in s. Maximize the length of each mini-tweet, so that each mini-tweet (except possibly the last) has a length equal to MAX_LENGTH.
    
    >>> s = 'code.org'
    >>> split_into_minitweets(s)
    ['code.org']
    >>> s = 'Do not just play on your phone - program it. - @BarackObama'
    >>> split_into_minitweets(s)
    ['Do not just', 'play on ', 'your phone', ' - program', 'it. - @BarackObama']
    """

# Sample solution 1:
tweets = []
for i in range(0, len(s), 10):
    tweets.append(s[i:i+10])
return tweets

# Sample solution 2:
tweets = []
while len(s) != 0:
    tweets.append(s[:MAX_LENGTH])
    s = s[MAX_LENGTH:]
return tweets
```