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(tweet2[2] + tweet1[-2])</td>
<td>no</td>
</tr>
<tr>
<td>print(tweet1[tweet2[-1]])</td>
<td>ERROR: index is str, not an int</td>
</tr>
<tr>
<td>print(tweet2[:tweet2.find('?')])</td>
<td>Want cheap snacks</td>
</tr>
<tr>
<td>print('cheap' in tweet2)</td>
<td>True</td>
</tr>
<tr>
<td>print('of' in tweet1 and 'of' in tweet2)</td>
<td>True</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.)

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

    # a list containing one str that starts with 2
    >>> mystery(['2'])
    '22'
    # a list containing three strings that start with 1, 3, and 5 respectively
    >>> mystery(['12', '34', '56'])
    '113355'
    ""

    result = ''
    for item in L:
        result = result + item[0] + item[0]

    return result
```

Page 1 of 5
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.

```python
def get_lowercase(s):
    """ (str) -> str
    Return all lowercase letters in s, or the empty string if there are no lowercase letters in s.

    >>> get_lowercase('123')
    ''
    >>> get_lowercase('a1b')
    'ab'
    >>> get_lowercase('1a2b3c4d')
    'abcd'
    ""
    i = len(s) - 1
    x = ''
    while i >= 0:
        if s[i].islower():
            x = s[i] + x
        i = i - 1
    return x
```
Question 4.  [5 marks]

Complete this function according to its docstring description.

```python
def hide_alnum(phrase):
    """ (str) -> str

    Return a new string that is the same as phrase but with each letter changed to the underscore symbol ('_') and each digit changed to the number sign symbol ('#').

    >>> hide_alnum('csc 108')
    '_-- ###'
    >>> hide_alnum('I h8 midterms!!!')
    '_ _# _________!!!'
    >>> hide_alnum('R2D2')
    '_#_#'
    """

    hidden = ''
    for ch in phrase:
        if ch.isalpha():
            hidden = hidden + '_'
        elif ch.isdigit():
            hidden = hidden + '#'
        else:
            hidden = hidden + ch

    return hidden
```

```python
def hide_alnum(phrase):
    """ (str) -> str

    Return a new string that is the same as phrase but with each letter changed to the underscore symbol ('_') and each digit changed to the number sign symbol ('#').

    >>> hide_alnum('csc 108')
    '_-- ###'
    >>> hide_alnum('I h8 midterms!!!')
    '_ _# _________!!!'
    >>> hide_alnum('R2D2')
    '_#_#'
    """

    hidden = ''
    for ch in phrase:
        if ch.isalpha():
            hidden = hidden + '_'
        elif ch.isdigit():
            hidden = hidden + '#'
        else:
            hidden = hidden + ch

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

Complete the function according to its docstring description.

```python
def get_at_symbol_indices(s):
    """ (str) -> list of int

    Return a list containing the index of each '@' in s.
    """

    >>> s = 'Username @uoft_cs is the Dept of Computer Science at UofT'
    >>> get_at_symbol_indices(s)
    [9]
    >>> s = 'See @UofTNews story on 2013 grad @TobiOgunbiyi & his startup @divnotes'
    >>> get_at_symbol_indices(s)
    [4, 33, 61]
    >>> s = '#UofT grad students #startup makes it easier for artists'
    >>> get_at_symbol_indices(s)
    []
    """

    # Sample solution 1:
    indices = []
    for i in range(len(s)):
        if s[i] == '@':
            indices.append(i)
    return indices

    # Sample solution 2:
    indices = []
    i = 0
    for ch in s:
        if ch == '@':
            indices.append(i)
            i = i + 1
    return indices

    # Sample solution 3:
    indices = []
    i = 0
    while i < len(s):
        if s[i] == '@':
            indices.append(i)
            i = i + 1
    return indices
```