**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><code>print('machine' in tweet1)</code></td>
<td>False</td>
</tr>
<tr>
<td><code>print(tweet2[2] + tweet1[-2])</code></td>
<td>no</td>
</tr>
<tr>
<td><code>print(tweet1[tweet2.find('W'):tweet2[-1]])</code></td>
<td>ERROR: stop index is type str, not int</td>
</tr>
<tr>
<td><code>print('Nobel' in tweet1 or 'ant' in tweet2)</code></td>
<td>True</td>
</tr>
<tr>
<td><code>print(not 'award' in tweet1)</code></td>
<td>False</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 has 'z' as its middle char
    >>> mystery(['z'])
    'z'
    # a list containing three strings that have 'j', 'e' and 't' as middle characters
    >>> mystery(['enjoy', 'every', 'question'])
    'jet'
    """

    result = ''
    for item in L:
        m = len(item) // 2
        result = result + item[m]

    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_last_letter(s):
    """ (str) -> str

    Return the last letter in s, or the empty string if there are no letters in s.
    
    >>> get_last_letter('abc')
    'c'
    >>> get_last_letter('1,2,3,4,...')
    ''
    """

    x = ''
    i = 0
    while i < len(s):
        if s[i].isalpha():
            x = s[i]
            i = i + 1
    return x
Question 4.  [5 marks]

When giving directions, we use the cardinal points on the compass: North, East, South and West. More precise directions can be described by longer compass descriptions. For example, NW stands for NorthWest, the direction that is halfway between North and West. NNW is halfway between North and NorthWest, and so on.

The opposite of a direction can be described by interchanging Ns and Ss, and interchanging Es and Ws. Complete this function according to its docstring description. Assume that the given direction is meaningful, and not something silly like 'NS' or 'NEW'.

```python
def opposite_direction(original):
    """ (str) -> str

    Precondition: each character in original is one of 'N', 'E', 'S' or 'W'.

    Return a new string with the direction opposite to original. If
direction is the empty string, return the empty string.

    >>> opposite_direction('NNE')
    'SSW'
    >>> opposite_direction('')
    ''
    """

    opposite = ''
    for ch in original:
        if ch == 'N':
            opposite = opposite + 'S'
        elif ch == 'E':
            opposite = opposite + 'W'
        elif ch == 'S':
            opposite = opposite + 'N'
        elif ch == 'W':
            opposite = opposite + 'E'

    return opposite
```

Page 3 of 4
**Question 5.**  [4 marks]

Complete the function according to its docstring description.

```python
def get_hash_symbol_indices(s):
    """ (str) -> list of int
    Return a list containing the index of each '#' in s.
    """
    indices = []
    for i in range(len(s)):
        if s[i] == '#':
            indices.append(i)
    return indices

# 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
```