This midterm consists of 5 questions on 6 pages (including this one). When you receive the signal to start, please make sure that your copy is complete.

- Comments are not required except where indicated, although they may help us mark your answers.
- No error checking is required: assume all user input and all argument values are valid.
- If you use any space for rough work, indicate clearly what you want marked.
- You may use a pencil; however, work written in pencil will not be considered for remarking.

# 1: _____/ 5
# 2: _____/ 2
# 3: _____/ 4
# 4: _____/ 5
# 5: _____/ 4
TOTAL: _____/20
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('machine' in tweet1)</td>
<td></td>
</tr>
<tr>
<td>print(tweet2[2] + tweet1[-2])</td>
<td></td>
</tr>
<tr>
<td>print(tweet1[tweet2.find('W'):tweet2[-1]])</td>
<td></td>
</tr>
<tr>
<td>print('Nobel' in tweet1 or 'ant' in tweet2)</td>
<td></td>
</tr>
<tr>
<td>print(not 'award' in tweet1)</td>
<td></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
    >>> mystery(                  )
    'z'
    >>> mystery(                  )
    '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.

```python
def (s):
    ""
    x = ''
    i = 0
    while i < len(s):
        if s[i].isalpha():
            x = s[i]
            i = i + 1
    return x
```

```python
""
```
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('')
    ''
    """
```
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.
    """
    >>> s = '#UofT grad students #startup makes it easier for artists'
    >>> get_hash_symbol_indices(s)
    [0, 20]
    >>> s = 'There are many #Harvard & #graduate #school #alumni on #Twitter'
    >>> get_hash_symbol_indices(s)
    [15, 26, 36, 44, 55]
    >>> s = '@UofTHacks do not worry, I will handle the twitter feeds!'
    >>> get_hash_symbol_indices(s)
    []
    """
```
Short Python function/method descriptions:

__builtins__:

int(x) -> int
   Convert x to an integer, if possible. A floating point argument will be truncated towards zero.

len(x) -> int
   Return the length of list, tuple, or string x.

print(value) --> NoneType
   Prints the values.

range([start], stop, [step]) -> list-like-object of int
   Return the integers starting with start and ending with stop - 1 with step specifying the amount to increment (or decrement). If start is not specified, the sequence starts at 0. If step is not specified, the values are incremented by 1.

str(x) -> str
   Return an object converted to its string representation, if possible.

str:

   x in s --> bool
      Produce True if and only if x is in s.

S.count(sub[, start[, end]]) --> int
   Return the number of non-overlapping occurrences of substring sub in string S[start:end]. Optional arguments start and end are interpreted as in slice notation.

S.find(sub[,i]) -> int
   Return the lowest index in S (starting at S[i], if i is given) where the string sub is found or -1 if sub does not occur in S.

S.isalpha() -> bool
   Return True if and only if all characters in S are alphabetic and there is at least one character in S.

S.isalnum() -> bool
   Return True if and only if all characters in S are alphanumeric and there is at least one character is S.

S.isdigit() -> bool
   Return True if and only if all characters in S are digits and there is at least one character in S.

S.islower() -> bool
   Return True if and only if all cased characters in S are lowercase and there is at least one cased character in S.

S.isupper() -> bool
   Return True if and only if all cased characters in S are uppercase and there is at least one cased character in S.

S.lower() -> str
   Return a copy of the string S converted to lowercase.

S.replace(old, new) -> str
   Return a copy of string S with all occurrences of the string old replaced with the string new.

S.upper() -> str
   Return a copy of the string S converted to uppercase.

list:

   x in L --> bool
      Produce True if and only if x is in list L

L.append(object) --> NoneType
   Append object to end of list L.

L.extend(iterable) --> NoneType
   Extend list L by appending elements from the iterable. Strings and lists are iterables whose elements are characters and list items respectively.