This midterm is double-sided, and consists of 5 questions and a list of function/method descriptions. 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.
- Do not remove any pages from the exam booklet.
- You may use a pencil; however, work written in pencil will not be considered for remarking.

# 1: _____/ 8  
# 2: _____/ 4  
# 3: _____/ 4  
# 4: _____/ 5  
# 5: _____/ 3  
TOTAL: _____/24
**Question 1.** [8 marks]

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

<table>
<thead>
<tr>
<th>Code</th>
<th>Output or Cause of Error</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>print(20 / 2 ** 2 + 3)</code></td>
<td></td>
</tr>
</tbody>
</table>
| `s = "hi there"`  
`print(s[3])` | |
| `values = [3, 7, 2]`  
`values.append(4)`  
`print(values)` | |
| `values = [1, 2, 3]`  
`values.extend(4)`  
`print(values)` | |
| `print('orange' > 'apple' or False)` | |
| `cloudy = True`  
`num = 0`  
`print(cloudy or 5 / num > 3)` | |
| `print(('3' + 3) == 6 and True)` | |
| `a_list = [13, 4, 7]`  
`b_list = a_list.append(5)`  
`print(a_list == b_list)` | |
Question 2. [4 marks]

In the function below, complete (i) the function description in the space provided, and (ii) the example function calls by adding arguments that result in the return values shown. (For the example calls, there may be several correct answers, and providing any one of them will earn full marks.)

```python
def mystery(message):
    """ (str) -> str

>>> mystery("
'hello'
>>> mystery(
'bb'
""

if len(message) < 5:
    return message[1::2]
else:
    return message
```
Question 3. [4 marks]

Read the function header and function body, and then complete the docstring. Write the type contract and the description, and give two examples that return different values. Preconditions are not required.

```python
def hidden_function_name(s):
    
    i = 0
    while i < len(s):
        char = s[i]
        if not (char.isalpha() or char.isdigit() or char == '_'):
            return False
        i = i + 1
    return True
```
Question 4. [5 Marks]

Complete this function according to its docstring description.

```python
def sum_of_digits(message):
    """ (str) -> int
    Return the sum of all digits that appear in message.
    """

>>> sum_of_digits('abc123')
6
>>> sum_of_digits('hello')
0
"""
Question 5.  [3 Marks]

Complete this function according to its docstring description.

def get_string_info(message, char, cutoff):
    """ (str, str, int) -> bool

    Precondition: len(char) == 1 and cutoff >= 0

    Return True if there are more occurrences of char in message
    than cutoff, and False otherwise.

    >>> get_string_info('hello world', 'l', 2)
    True
    >>> get_string_info('apple', 'p', 2)
    False
    """

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 string 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.