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.

Code	Output or Cause of Error
`print(20 / 2 ** 2 + 3)`	`8.0`
`s = "hi there"` `print(s[3])`	`t`

`values = [3, 7, 2]`
`values.append(4)`
`print(values)`
	`[3, 7, 2, 4]`

`values = [1, 2, 3]`
`values.extend(4)`
`print(values)`
	`ERROR - argument must be iterable (ie a list)`

`print('orange' > 'apple' or False)`	`True`

`cloudy = True`
`num = 0`
`print(cloudy or 5/num > 3)`
	`True`

`print(('3' + 3) == 6 and True)`	`ERROR - can't add str and int`

`a_list = [13, 4, 7]`
`b_list = a_list.append(5)`
`print(a_list == b_list)`
	`False`

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.)

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

    Return a string containing every odd indexed character if message is
    shorter than 5 characters, otherwise return message.

    >>> mystery('hello')
    'hello'
    >>> mystery('abab')
    '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.

```
def hidden_function_name(s):
    """ (str) -> bool

    Return True iff s contains only letters, digits, and underscores.

    >>> hidden_function_name('banana_phone1')
    True
    >>> hidden_function_name('hello!')
    False
    """

    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.

```
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
   """


   total = 0
   for char in message:
      if char.isdigit():
         total = total + int(char)
   return total
```

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
    """

    return message.count(char) > cutoff
```