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
`message = 'Hi Jacqueline'` `print(message[5])`	`c`
`print(2 * 16 / 2 ** 2 + 1)`	`9.0`
`happy = True` `print(happy or 5 / 0 == 1)`	`True`
`print(8 == 3 + '5' and True)`	`ERROR - can't concatenate int and str`

`cats = ['Mittens', 'Socks']`
`more_cats = cats.append('Milo')`
`print(more_cats)`
	`None`

`total = 0`
`for i in range(1, 4):`
`    total = total + i`
`print(total)`
	`6`

`a = [1.5, 2]`
`a.extend(4)`
`print(len(a) == 3)`
	`ERROR - can't extend with int`

`s = 'hello'`
`s[0] = 'j'`
`print(s)`
	`ERROR - can't modify str`

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.) You do not need to add any preconditions.

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

    Return the number of elements of values that start with an uppercase letter.

    >>> mystery(['A', 'b', 'C'])
    2
    >>> mystery(['hello'])
    0
    """

    count = 0

    for item in values:
        if item[0].isupper():
            count = count + 1

    return count
```

Question 3. [4 marks]

Consider the following two function definitions (docstrings excluded due to space). Beside each code fragment in the table below, write what is printed when the code fragment is executed.

```
def first(value):
    total = 0
    if value > 10:
        total = total + 10
    elif value < 5:
        total = total + 5
    else:
        total = total + 1
    return total

def second(value):
    total = 0
    if value > 10:
        total = total + 10
    if value < 5:
        total = total + 5
    else:
        total = total + 1
    return total
```

Code	Output
`print(first(1))`	5
`print(second(2))`	5
`print(first(12))`	10
`print(second(12))`	11

Question 4. [5 marks]

For our purposes, a phone number is a string of 10 or more characters, and contains only digits, spaces, and dashes (`'-'`).

Complete the body of the `is_valid_phone_number` function by filling in the boxes below.

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

    Return True iff s is a valid phone number.

    >>> is_valid_phone_number('416 123-4567')
    True
    >>> is_valid_phone_number('416 EAT-CAKE')
    False
    >>> is_valid_phone_number('44 1908 640404')
    True
    >>> is_valid_phone_number('416 978')
    False
    """

    if len(s) < 10:
        return False
    i = 0
    while i < len(s):
        if not (s[i].isdigit() or s[i] in '- '):
            return False
        i = i + 1
    return True
```

Question 5. [3 marks]

Complete this function according to its docstring description.

```
def has_even_num_of_char(s, ch):
    """ (str, str) -> bool

    Precondition: s contains at least one occurrence of ch

    Return True iff s contains an even number of the character ch.

    >>> has_even_num_of_char('hello', 'l')
    True
    >>> has_even_num_of_char('hello', 'e')
    False
    """



    return s.count(ch) % 2 == 0

    # OR loop with counter
    count = 0
    for c in s:
        if c == ch:
            count = count + 1
    return count % 2 == 0
```