Question 1. [4 marks]

Beside each code fragment in the table below, give the output. If the code would cause an error, write ERROR and give a brief explanation.

Code	Output or Cause of Error

`toys = ['ball', 'car', 'doll']`
`stuff = toys[0:2]`
`stuff[0] = 'phone'`
`print(toys)`
	`['ball', 'car', 'doll']`

`toys = ['ball', 'car', 'doll']`
`stuff = toys`
`stuff.pop()`
`print(toys)`
	`['ball', 'car']`

`days = [['Mon', 'Oct', 21], ['Wed', 'Nov', 7],`
`['Fri', 'Dec', 6]]`
`print(days[1][1][1:2])`
	`o`

`days = [['Mon', 'Oct', 21], ['Wed', 'Nov', 7],`
`['Fri', 'Dec', 6]]`
`print(days[-1][-2])`
	`Dec`

`days = [['Mon', 'Oct', 21], ['Wed', 'Nov', 7],`
`['Fri', 'Dec', 6]]`
`print(days[0][0] == 'Thurs' and days[4] == [])`
	`False`

`days = [['Mon', 'Oct', 21], ['Wed', 'Nov', 7],`
`['Fri', 'Dec', 6]]`
`print(days[-1][1:])`
	`['Dec', 6]`

Question 2. [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 all_same_letter(s):
    """ (str) -> bool
    Return True iff all the characters of s are the same disregarding case.

    >>> all_same_letter('aaAaaAA')
    True
    >>> all_same_letter('9999')
    True
    >>> all_same_letter('Zzzz Zzzz')
    False

    """
    if len(s) == 0:
        return True

    first_lower = s[0].lower()
    for ch in s:
        if ch.lower() != first_lower:
            return False

    return True
```

Question 3. [3 marks]

Complete the function below according to its docstring.

```
def make_absolute(nums):
    """ (list of number) -> Nonetype

    Replace each item in nums with its absolute value.

    >>> nums = [1, -4.5, 0.2, -6]
    >>> make_absolute(nums)
    >>> nums
    [1, 4.5, 0.2, 6]
    """

    for i in range(len(nums)):
        nums[i] = abs(nums[i])
```

Question 4. [4 marks]

Complete the function below according to its docstring.

```
def new_math_test(old_test, operators):
    """ (str, str) -> str

    Return a copy of old_test with the following changes:
      All digits are replaced by ' '.
      All symbols (not letters, digits, or spaces) in operators
      are replaced by '_'.
      All other characters are left the same.

    >>> new_math_test('x^3 + 3x^2 - 17', '+-')
    'x^  _  x^  _   '
    >>> new_math_test('Find the integral of 4x*e^2x + 19', '')
    'Find the integral of  x*e^ x +   '
    """

    new_test = ''
    for c in old_test:
        if c.isdigit():
            new_test += ' '
        elif c in operators:
            new_test += '_'
        else:
            new_test += c

    return new_test
```

Question 5. [5 marks]

Two parties are running in many elections in regions throughout a country. In each region, whichever party receives the most votes wins the region, and whichever party wins the most regions will get to form the government.

Complete the following function according to the description above and the docstring below.

```
def election_results(party1_votes, party2_votes):
    """ (list of int, list of int) -> list of int

    Pre-condition: len(party1_votes) == len(party2_votes)

    Return a list of integers representing the number of regions won by each party. The
    first element is the number of regions won by party 1, the second is the number won by
    party 2 and the third is the number of ties. The number of votes for each region are
    in the lists party1_votes and party2_votes, with one entry per region.

    >>> election_results([5, 2, 8], [0, 0, 9])
    [2, 1, 0]
    >>> election_results([17, 13, 40, 100], [18, 10, 40, 0])
    [2, 1, 1]
    """

    wins1 = 0
    wins2 = 0
    ties = 0
    for i in range(len(party1_votes)):
        if party1_votes[i] > party2_votes[i]:
            wins1 += 1
        elif party1[i] < party2[i]:
            wins2 += 1
        else:
            ties += 1

    return [wins1, wins2, ties]
```