Problem 1

Complete the function below that takes a Python list as an argument and returns its second element. Example usage:

`get_2nd_element([0, 20, 30, 40])`

will return `20`.

In [ ]:
```
def get_2nd_element(alist):

    #return #2nd element of list
    return alist[1]
```

In [ ]:
```
alist = [0, 20, 30, 40]
get_2nd_element(alist)
```

Out[ ]: 20

Problem 2

Complete the function below. The function takes a single tuple as an argument. The tuple should have exactly 2 elements and should return the sum of those 2 elements. In the event that the user supplies more than 2 elements, the function should return the string `"Tuple must have exactly 2 elements!"`. Example usage:

`add_tuple_elements((2, 3))`

should return `5`.

And

`add_tuple_elements((2, 3, 6))`

should return `"Tuple must have exactly 2 elements!"`

Hint: You can determine how many elements are in a tuple (or a list) with the Python function `len()`.

In [ ]:
```
def add_tuple_elements(atuple):

    #return #The sum of the tuple elements or "Tuple must have exactly 2 elements!"
    if len(atuple)>2:
        return "Tuple must have exactly 2 elements!"
    else:
        return atuple[0] + atuple[1]
```

In [ ]:
```
add_tuple_elements((2, 3))
```

Out[ ]: 5

In [ ]:
```
add_tuple_elements((2, 3, 6))
```

Out[ ]: 'Tuple must have exactly 2 elements!'

Problem 3

Complete the function below that takes a Python dictionary as an argument. Given a dictionary of the form:

`adict = {'name': 'Romeo Montague', 'age': 32, 'DOB': '01/21/1594'}`

the function should return the `'age'` value and `'DOB'` value as a single tuple. Example usage:

`get_age_and_dob(adict)`

should return `(32, '01/21/1594')`

In [ ]:
```
def get_age_and_dob(adict):
    return str(adict['age']) + ', ' + adict['DOB']
```

In [ ]:
```
adict = {'name': 'Romeo Montague', 'age': 32, 'DOB': '01/21/1594'}
```

In [ ]:
```
get_age_and_dob(adict)
```

Out[ ]: '32, 01/21/1594'