

**Quiz 10****Name:** \_\_\_\_\_

1. Draw the memory state diagram for the following program at the point of time when the program reaches line 6 ( remember you have not yet come out of the function `perform_magic`):

```
1 def perform_magic(a, b):  
2     tmp = a  
3     a = b  
4     tmp.append('carp')  
5     b = []  
6     # How does the memory state diagram look here?  
7  
8 a = ['apple']  
9 b = ['amy']  
10 perform_magic(a, b)
```

**Answer:**

2. [ 4 marks ] Examine each of the following snippets of code and list the output expected. Write '- error -' if you think the code will crash (i.e. raise an error) during execution. Write '- nothing -' if you think the code will not produce any output.

```
is_mystery = False
is_magic = False

for i in range(3):
    if is_mystery:
        print('apple')
        is_magic = not is_magic

    if is_mystery and is_magic:
        print('pear')
    elif is_magic:
        print('orange')

    is_mystery = not is_mystery
```

3. [ 3 marks ] Implement the following functions:

- is\_perfect\_square
- get\_perfect\_squares

Any number which can be expressed as the product of two whole equal numbers is classified as a perfect square. For example, 64 can be written as  $8 * 8$  hence 64 is a perfect square. To check if a number is a perfect square, do the following:

- Compute the square root of the given number.
- Add 0.5 to the square root and then drop all decimal places to get a whole number, square\_root.
- Square the number, i.e. square\_root \* square\_root

If the number is 4,

- the square root is 1.999999 (since floating-point values are never precise)
- $1.999999 + 0.5$  to get 2.4999000000000002. Drop all decimal places to get 2.
- $2 * 2$  is 4. Since we got back the original value of 4, the number is a perfect square.

**Hint:** You will need to use the `sqrt` function from the `math` module to obtain the square root of a float value. For example, given the following script:

```
from q2 import get_perfect_squares

print("Test 1")
result = get_perfect_squares([1, 2, 3, 4, 5])
print("Expected:[1, 4]")
print(f"Actual   :{result}")
print()
```

```
print("Test 2")
result = get_perfect_squares([9, 7, 7, 4, 5])
print("Expected:[9, 4]")
print(f"Actual   :{result}")
print()

print("Test 3")
input = [2, 6, 8, 12, 14]
result = get_perfect_squares(input)
print("Expected:[] [2, 6, 8, 12, 14]")
print(f"Actual   :{result} {input}")
print()

print("Test 4")
result = get_perfect_squares([])
print("Expected:[]")
print(f"Actual   :{result}")
print()

print("Test 5")
result = get_perfect_squares([4])
print("Expected:<class 'list'> <class 'int'>")
print(f"Actual   :{type(result)} {type(result[0])}")
print()
```

It will generate the following output:

```
Test 1
Expected:[1, 4]
Actual   :[1, 4]

Test 2
Expected:[9, 4]
Actual   :[9, 4]

Test 3
Expected:[] [2, 6, 8, 12, 14]
Actual   :[] [2, 6, 8, 12, 14]

Test 4
Expected:[]
Actual   :[]

Test 5
Expected:<class 'list'> <class 'int'>
Actual   :<class 'list'> <class 'int'>
```

```
# Answer
```

```
# Include any necessary imports here
```

```
def is_perfect_square(number):
```

```
    '''
```

```
    Checks if the number is a perfect square
```

```
    Parameter/Argument:
```

```
        number (type: int): the value that we are checking
```

```
    Returns:
```

```
        True if the number is a perfect number. False otherwise.
```

```
    Examples:
```

```
        1. If the argument is 4, this function returns True.
```

```
        2. If the argument is 7, this function returns False.
```

```
    '''
```

```
def get_perfect_squares(values):  
    '''  
    Returns a new list that contains all numbers from values that  
    are perfect squares.  
  
    Parameter/Argument:  
        values (type: list): a list of positive numbers  
  
    Returns:  
        a new list of all the perfect numbers from  
  
    Examples:  
    If the argument is [1, 4, 5], this function returns [1, 4].  
    (1 * 1 = 1 and 2 * 2 = 4)  
    '''
```