Quiz 9 Name: _____

1. **[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.

Code	Output

`# Part 1`
`def test_one(x):`
`  print('alpha')`
`  return x > 4`
 	
`def test_two(x):`
`  print('beta')`
`  return x <= 8`
 	
`for i in range(4, 6):`
`  if test_one(i) and test_two(i):`
`  print("apple")`
`  elif test_one(i) or test_two(i):`
`  print("orange")`
`  print('--')`	

2. **[3 marks]** Given that the variables `x`, `y` and `z` are of type `int`, and `ch` is of type `char`. Write a new boolean expression that is the **NEGATION** of each of the following Boolean expressions that is simplified. You need to apply De Morgan's laws to simplify the expression.

	Expression	Negation of Expression
(e.g.)	`x > 1`	Acceptable: `x <= 1` Not acceptable: `not (x > 1)`
(a)	`not (x < 5) and (y >= 7)`	
(c)	`ch < '0' or ch > '9'`	
(d)	`x > 2 and y > 3 or z > 4`	

3. **[3 marks]** Implement the `sort_odd_even` function. The function takes in an array of non-negative integers, `numbers` and returns an array consisting of all the even elements of `numbers`, followed by all the odd elements of `numbers`. You **MUST** preserve the order the digit appears in `numbers`.

   For example, given the following script:

```
from q2 import sort_odd_even

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

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

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

print("Test 4")
result = sort_odd_even([9, 5, 1])
print("Expected:[9, 5, 1]")
print(f"Actual  :{result}")
print()

print("Test 5")
result = sort_odd_even([1, 2, 3, 4, 5])
print("Expected:<class 'list'> <class 'int'>")
print(f"Actual  :{type(result)} {type(result[0])}")
print()
```

It will generate the following output:

```
Test 1
Expected:[2, 4, 1, 3, 5]
Actual  :[2, 4, 1, 3, 5]

Test 2
Expected:[4, 9, 7, 7, 5]
Actual  :[4, 9, 7, 7, 5]

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

Test 4
Expected:[9, 5, 1]
Actual  :[9, 5, 1]

Test 1
Expected:<class 'list'> <class 'int'>
Actual  :<class 'list'> <class 'int'>
```

```
# Answer
```