Started on	Sunday, 16 August 2020, 10:22 PM
State	Finished
Completed on	Monday, 17 August 2020, 6:01 PM
Time taken	19 hours 38 mins
Grade	**9.80** out of 10.00 (**98**%)

Question **1**

Correct

Mark 0.50 out of 0.50

For the following block of code, select the most appropriate run-time formula in terms of primitive operations needed and the time complexity for input of size n:

```
def rate(n):
    i = n
    total = 0
    while i > 0:
        total += i
        i -= 1
    return total
```

Select one:

- a. f(n) = 2n, time complexity = O(n)
- b. f(n) = 3n+ 4, time complexity = O(n) ✔
- c. f(n) = 2n + 4, time complexity = O(n)
- d. f(n) = n + 2, time complexity = O(n)
- e. f(n) = 2n + 2, time complexity = O(n)

Your answer is correct.

Note: Time Complexity of a loop is considered as O(n) if the loop variables is incremented / decremented by a constant amount.

Correct

Marks for this submission: 0.50/0.50.

Question **2**

Correct

Mark 0.45 out of 0.50

For the following block of code, select the most appropriate run-time formula in terms of primitive operations needed and the time complexity for input of size n:

```
def rate(n):
    i = n
    total = 0
    while i > 1:
        total += i
        i //= 2
    return total
```

Select one:

- a. f(n) = log(n) + 4, time complexity = O(log n)
- b. f(n) = 3*n + 4, time complexity = O(n)
- c. f(n) = 3*log(n) + 4, time complexity = O(log n) ✔
- d. f(n) = 3 * (n/2) + 4, time complexity = O(n)
- e. f(n) = n + 4, time complexity = O(n)

Your answer is correct.

Time Complexity of a loop is considered as O(Log n) if the loop variables is divided / multiplied by a constant amount.

Correct

Marks for this submission: 0.50/0.50. Accounting for previous tries, this gives **0.45/0.50**.

Question **3**

Correct

Mark 0.40 out of 0.50

For the following block of code, select the most appropriate Big-O running time for input of size n:

```
def rate(n):
    i = 0
    total = 0
    while i < 10:
        j = 0
        while j < 10:
            total += j
            j += 1
        i += 1
    return total
```

Select one:

- a. O(n)
- b. O(log n)
- c. O(1) ✔
- d. None of the others
- e. O(n log n)

Your answer is correct.

Correct

Marks for this submission: 0.50/0.50. Accounting for previous tries, this gives **0.40/0.50**.

Question **4**

Correct

Mark 0.50 out of 0.50

Consider the following function:

```
def rate(n):
    i = 0
    total = 0
    while i < 10:
        j = 0
        while j < n:
            total += j
            j += 1
        i += 1
    return total
```

Count the number of operations that would be executed. Your output should be in the format:

```
Number of operations: <count>
```

where <count> is an integer value.

Answer: (penalty regime: 0, 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 %)

Reset answer

Test	Expected Result
rate(1)	Number of operations: 74
rate(10)	Number of operations: 344
rate(100)	Number of operations: 3044

Add row Delete row

	The function being tested	Function is correct?	Passes all your tests?	

Passed all tests! ✔

Correct

Marks for this submission: 0.50/0.50.

Question **5**

Correct

Mark 0.50 out of 0.50

For the following block of code, select the most appropriate Big-O running time for input of size n:

```
def rate(n):
    i = 0
    total = 0
    while i < n:
        j = 0
        while j < 10:
            total += j
            j += 1
        i += 1
    return total
```

Select one:

- a. O(n^2)
- b. O(log n)
- c. None of the others
- d. O(n) ✔
- e. O(1)

Your answer is correct.

Correct

Marks for this submission: 0.50/0.50.

Question **6**

Correct

Mark 1.00 out of 1.00

Consider the following function:

```
def rate(n):
    total = 0
    i = 0
    while i < n:
        j = 0
        while j < n:
            total += j
            j += 1
        i += 1
    return total
```

Modify the function so it will print out the total number of statements that are executed inside the rate function shown. Do not count any additional statements that you add to the code when you modify it. You should count the lines containing loop conditions as being executed each time the condition is checked. Note that a loop condition is checked 1 time more than the loop body is executed. Your output should be in the format:

```
Number of operations: <count>
```

where <count> is an integer value.

For example:

Test	Result
`rate(2)`	`Number of operations: 24`

Answer: (penalty regime: 0, 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 %)

```
def rate(n):
    c = 4
    total = 0
    i = 0
    while i < n:
        c += 4
        j = 0
        while j < n:
            c += 3
            total += j
            j += 1
        i += 1
    return print("Number of operations: {}".format(c))
```

	Test	Expected	Got	
✔	`rate(2)`	`Number of operations: 24`	`Number of operations: 24`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **7**

Correct

Mark 1.00 out of 1.00

Consider the following function:

```
def rate(n):
    total = 0
    i = 1
    while i < n:
        j = 0
        while j < n:
            total += j
            j += 1
        i *= 2
    return total
```

Modify the function so it will print out the **total number of operations** that are executed inside the `rate()` function shown. Do not count any additional statements that you add to the code when you modify it. Your output should be in the format:

```
Number of operations: <count>
```

where <count> is an integer value.

For example:

Test	Result
`rate(2)`	`Number of operations: 14`

Answer: (penalty regime: 0, 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 %)

```
def rate(n):
    c = 4
    total = 0
    i = 1
    while i < n:
        c += 4
        j = 0
        while j < n:
            c += 3
            total += j
            j += 1
        i *= 2
    return print("Number of operations: {}".format(c))
```

	Test	Expected	Got	
✔	`rate(2)`	`Number of operations: 14`	`Number of operations: 14`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **8**

Correct

Mark 0.45 out of 0.50

For the following block of code, select the most appropriate run-time formula in terms of primitive operations needed and the time complexity for input of size n:

```
def rate(n):
    i = 1
    total = 0
    while i < n:
        j = 0
        while j < n:
            total += j
            j += 1
        i *= 2
    return total
```

Select one:

- a. f(n) = 3*log(n) + 4, time complexity = O(log n)
- b. f(n) = (3/2)*n^2+ 2n + 4, time complexity = O(n^2)
- c. f(n) = 4 * (n^2) + 4, time complexity = O(n^2)
- d. f(n) = 34n + 4, time complexity = O(n)
- e. f(n) = 3n * log n + 4 log n + 4, time complexity = O(n log n) ✔

Your answer is correct.

Correct

Marks for this submission: 0.50/0.50. Accounting for previous tries, this gives **0.45/0.50**.

Question **9**

Correct

Mark 1.00 out of 1.00

Consider the following function:

```
def rate(n):
    total = 0
    i = 0
    while i < n:
        j = 0
        while j < 2 * n:
            total += j
            j += 1
        i += 1
    return total
```

Modify the function so it will print out the total number of statements that are executed inside the rate function shown. Do not count any additional statements that you add to the code when you modify it. You should count the lines containing loop conditions as being executed each time the condition is checked. Note that a loop condition is checked 1 time more than the loop body is executed. Your output should be in the format:

```
Number of operations: <count>
```

where <count> is an integer value.

For example:

Test	Result
`rate(2)`	`Number of operations: 36`

Answer: (penalty regime: 0, 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 %)

```
def rate(n):
    c = 4
    total = 0
    i = 0
    while i < n:
        c += 4
        j = 0
        while j < 2 * n:
            c += 3
            total += j
            j += 1
        i += 1
    return print("Number of operations: {}".format(c))
```

	Test	Expected	Got	
✔	`rate(2)`	`Number of operations: 36`	`Number of operations: 36`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **10**

Correct

Mark 0.50 out of 0.50

For the following block of code, select the most appropriate run-time formula in terms of primitive operations needed and the time complexity for input of size n:

```
def rate(n):
    i = 0
    total = 0
    while i < n:
        j = 0
        while j < 2 * n:
            total += j
            j += 1
        i += 1
    return total
```

Select one:

- a. f(n) = 34n + 4, time complexity = O(n)
- b. f(n) = 6* (n^2) + 4, time complexity = O(n^2)
- c. f(n) = 3*log(n) + 4, time complexity = O(log n)
- d. f(n) = 6*n^2+ 4n + 4, time complexity = O(n^2) ✔
- e. f(n) = 3*n^2+ 4n + 4, time complexity = O(n^2)

Your answer is correct.

Correct

Marks for this submission: 0.50/0.50.

Question **11**

Correct

Mark 1.00 out of 1.00

Consider the following function:

```
def rate(n):
    total = 0
    i = 0
    while i < 4:
        j = 0
        while j < i:
            total += j
            j += 1
        i += 1
    return total
```

Modify the function so it will print out the total number of statements that are executed inside the rate function shown. Do not count any additional statements that you add to the code when you modify it. You should count the lines containing loop conditions as being executed each time the condition is checked. Note that a loop condition is checked 1 time more than the loop body is executed. Your output should be in the format:

```
Number of operations: <count>
```

where <count> is an integer value.

For example:

Test	Result
`rate(2)`	`Number of operations: 38`

Answer: (penalty regime: 0, 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 %)

```
def rate(n):
    c = 4
    total = 0
    i = 0
    while i < 4:
        c += 4
        j = 0
        while j < i:
            c += 3
            total += j
            j += 1
        i += 1
    return print("Number of operations: {}".format(c))
```

	Test	Expected	Got	
✔	`rate(2)`	`Number of operations: 38`	`Number of operations: 38`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **12**

Correct

Mark 0.50 out of 0.50

For the following block of code, select the most appropriate Big-O running time for input of size n:

```
def rate(n):
    i = 0
    total = 0
    while i < n:
        j = 0
        while j < i:
            total += j
            j += 1
        i += 1
    return total
```

Select one:

- a. O(n^3)
- b. O(log n)
- c. O(n log n)
- d. O(n^2) ✔
- e. O(n)

Your answer is correct.

Correct

Marks for this submission: 0.50/0.50.

Question **13**

Correct

Mark 1.00 out of 1.00

For the following block of code, select the most appropriate Big-O running time for input of size n:

```
def find_difference(data):
    largest = data[0]
    smallest = data[0]
    for x in data:
        if largest < x:
            largest = x
        if smallest > x:
            smallest = x
    return largest - smallest
```

Select one:

- a. None of the others
- b. O(log n)
- c. O(n^2)
- d. O(n) ✔
- e. O(1)

Your answer is correct.

Correct

Marks for this submission: 1.00/1.00.

Question **14**

Correct

Mark 1.00 out of 1.00

For the following block of code, select the most appropriate Big-O running time for input of size n:

```
def no_evens(numbers):
    for x in numbers:
        if x %2 == 0:
            return False
    return True
```

Select one:

- a. O(n^2)
- b. O(1)
- c. O(n) ✔
- d. None of the others
- e. O(log n)

Your answer is correct.

Correct

Marks for this submission: 1.00/1.00.

◄ Lab04 (1%) - Exceptions

Jump to...

Lab07 (1%) - Test Revision ►