Big O Notation

Practice Questions

Question 1

Complete the table below to indicate the notation used to describe each of the following BigO notations:

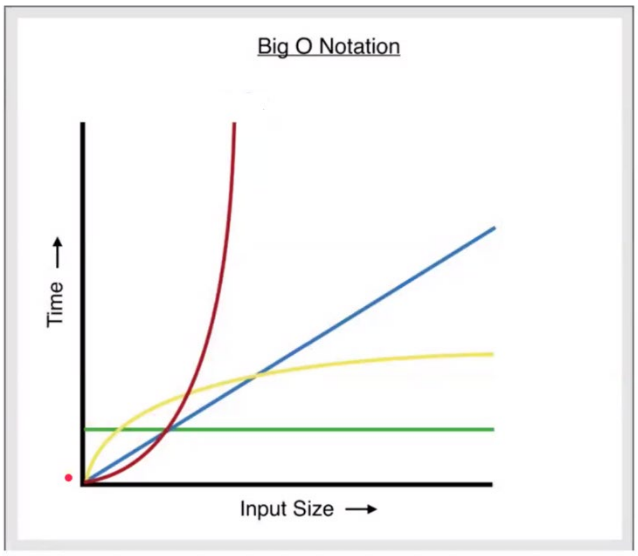
Notation	**Name**
	Constant
	Exponential
	Linear
	Logarithmic
	Quadratic (or polynomial)

Question 2:

Put the list from above in order from least complex/efficient to most complex/efficient.

Question 3:

Label the graph below with a key to indicate what each coloured line represents



Question 4:

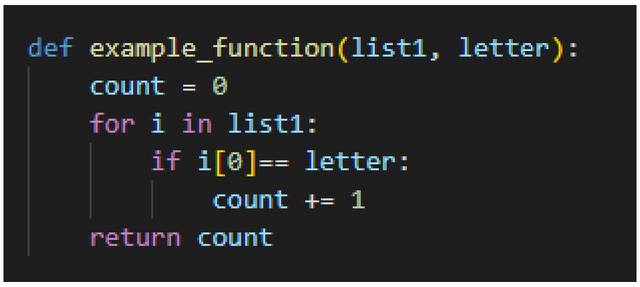
Which case scenario does Big O use:

a. Best case
b. Worst case
c. Average case
d. All cases must be considered

Question 5:

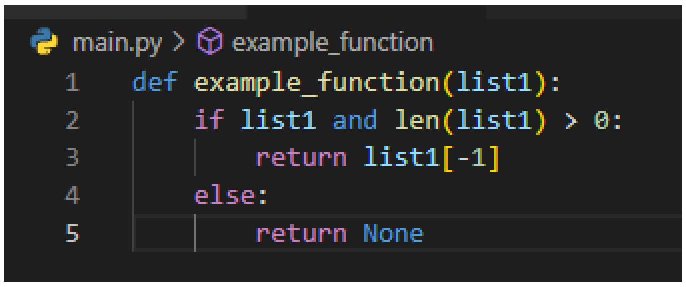
What is the complexity of the following functions:

a.



```python
def example_function(list1, letter):
    count = 0
    for i in list1:
        if i[0]== letter:
            count += 1
    return count
```

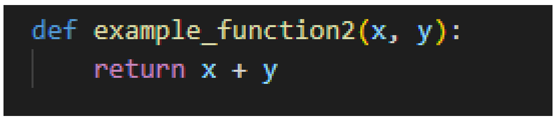
b.



main.py > example_function

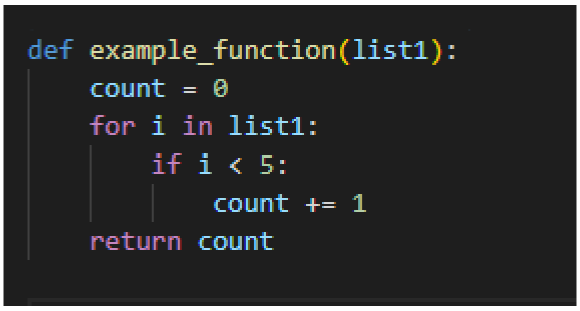
```python
def example_function(list1):
    if list1 and len(list1) > 0:
        return list1[-1]
    else:
        return None
```

c.



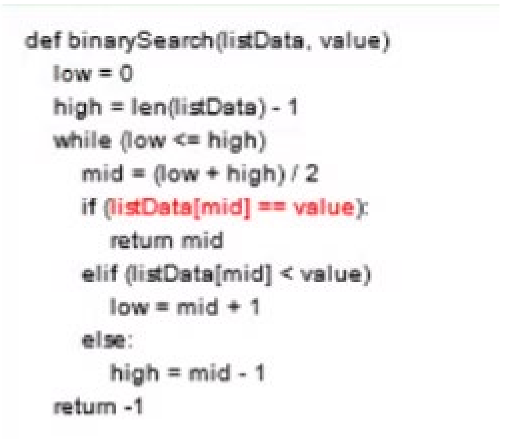
```python
def example_function2(x, y):
    return x + y
```

d.



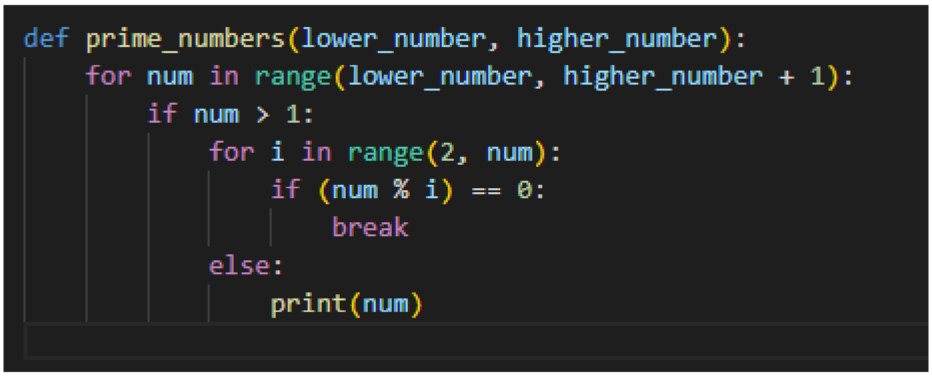
```python
def example_function(list1):
    count = 0
    for i in list1:
        if i < 5:
            count += 1
    return count
```

e.



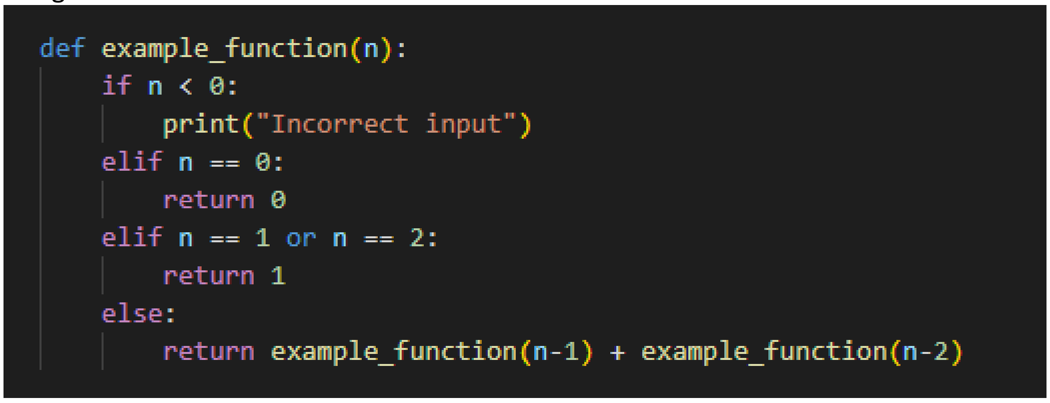
```
def binarySearch(listData, value)
    low = 0
    high = len(listData) - 1
    while (low <= high)
        mid = (low + high) / 2
        if (listData[mid] == value):
            return mid
        elif (listData[mid] < value)
            low = mid + 1
        else:
            high = mid - 1
    return -1
```

f.



```python
def prime_numbers(lower_number, higher_number):
    for num in range(lower_number, higher_number + 1):
        if num > 1:
            for i in range(2, num):
                if (num % i) == 0:
                    break
            else:
                print(num)
```

g.



```python
def example_function(n):
    if n < 0:
        print("Incorrect input")
    elif n == 0:
        return 0
    elif n == 1 or n == 2:
        return 1
    else:
        return example_function(n-1) + example_function(n-2)
```

Question 6:

Rewrite the following logarithms so that they are expressed as a quadradic:

$\log_3 81 = 4$
$\log_2 32 = 5$
$\log_2 128 = 7$
$\log_3 243 = 5$

Question 6

Calculate the following logarithms:

$\log_2 8$
$\log_2 64$
$\log_2 128$
$\log_2 1$
$\log_2 32$
$\log_3 125$

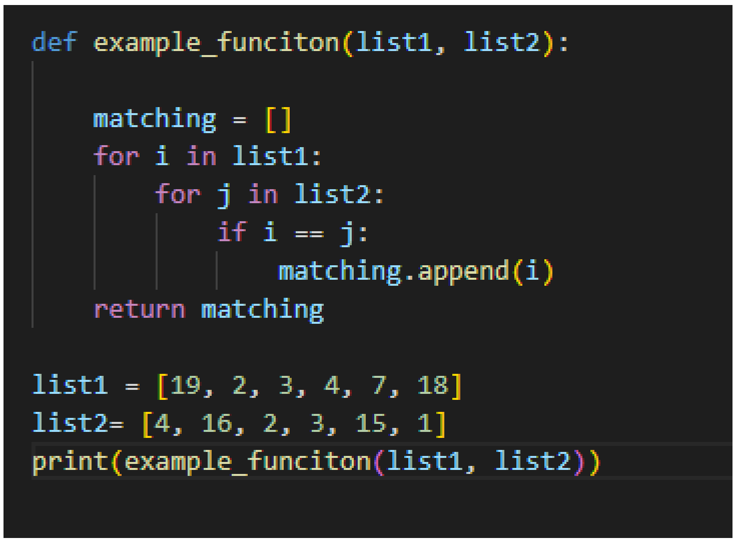
Question 7

Given an array of n, how long will a binary search take to find a specific value assuming the data is sorted. In the following cases:

a) N= 15
b) N= 25
c) N= 50
d) N=100
e) N= 1000

Question 8

The code snippet below relates to the questions which follow:



```python
def example_funciton(list1, list2):

    matching = []
    for i in list1:
        for j in list2:
            if i == j:
                matching.append(i)
    return matching

list1 = [19, 2, 3, 4, 7, 18]
list2= [4, 16, 2, 3, 15, 1]
print(example_funciton(list1, list2))
```

a) What is the complexity of the functions?
b) How many times does the outer loop of the function run?
c) How many times does the inner loops of the function run on each iteration in the worst case scenario?
d) In the worst-case scenario how many steps will this take?
e) What could the complexity of the function be reduced to if the items in list2 were sorted in order from smallest to largest?