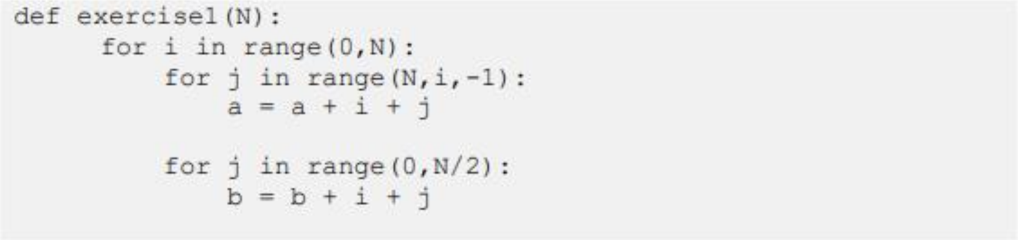
Q 3 – Complexity [25 Points]

b) [15 points] Give the worst case Big-O runtime of code snippets given in part (a) – (e)? Explain your working.

a)



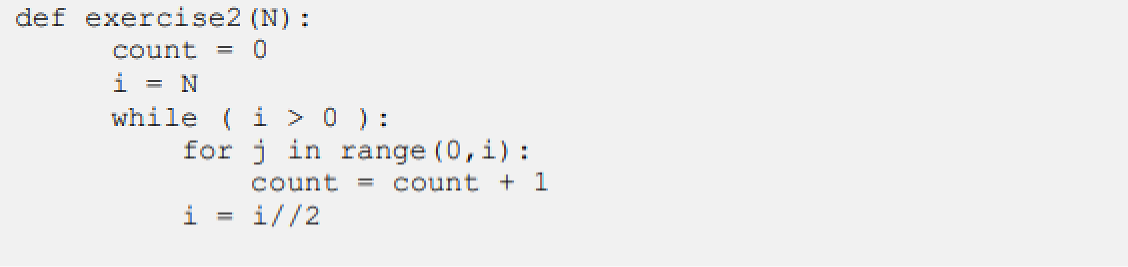
```
def exercise1(N):
      for i in range(0,N):
          for j in range(N,i,-1):
              a = a + i + j

          for j in range(0,N/2):
              b = b + i + j
```

The parent for loop = N
First inner for loop = (N+1)/2
Second inner for loop = N/2

N * [(N+1)/2 + N/2]
N^2 + N/2 (ignore the less significant)

O(N^2)

b)

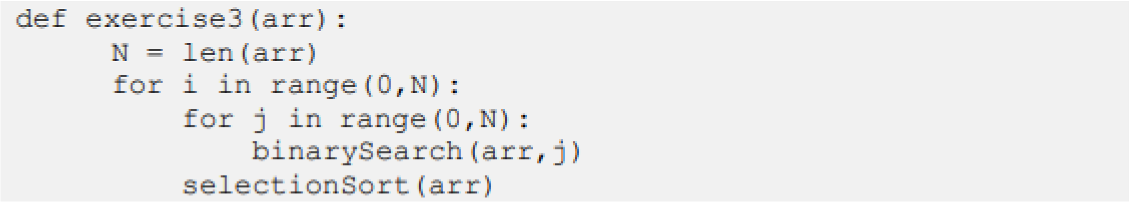


```
def exercise2(N):
      count = 0
      i = N
      while ( i > 0 ):
          for j in range(0,i):
              count = count + 1
          i = i//2
```

Using the Geometric Series formula we can deduce
N(1/(1-(1/2))) = 2N

O(N)

c)



```
def exercise3(arr):
    N = len(arr)
    for i in range(0,N):
        for j in range(0,N):
            binarySearch(arr,j)
        selectionSort(arr)
```

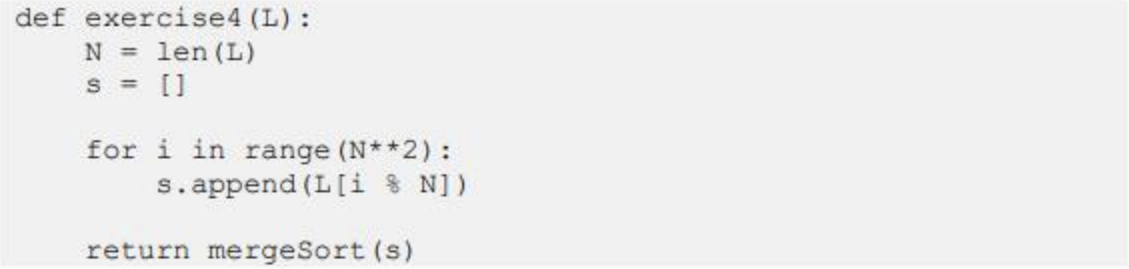
Complexity of Binary Search = O(logN)
Complexity of Selection Sort = O(N^2)

Parent for-loop = N
    -Inner for loop = N
        -Inner Binary Search = logN
    -Selection Sort = N^2

N * (NlogN + N^2)
N^3 + (N^2)(logN) -ignore the less significant

O(N^3)

d)



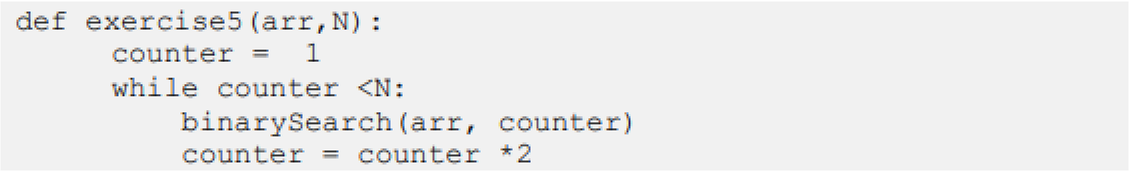
```
def exercise4(L):
    N = len(L)
    s = []

    for i in range(N**2):
        s.append(L[i % N])

    return mergeSort(s)
```

For-loop = N^2
Mergesort = O(NlogN)

N^2 + NlogN (ignore the insignificant)

O(N^2)

e)



```
def exercise5(arr,N):
    counter =  1
    while counter <N:
        binarySearch(arr, counter)
        counter = counter *2
```

While loop = logN
Binary Search = logN

logN * logN

O(logN)^2