B.Sc. Engg. CSE 8th Semester

ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)

ORGANISATION OF ISLAMIC COOPERATION (OIC)

Department of Computer Science and Engineering (CSE)

CSE 4810: Algorithm Engineering Lab

1. For each group of functions, sort the functions in increasing order of asymptotic(big-O) complexity:
   a) $f_1(n) = n^{0.9999999} \log(n^{2019})$
   b) $f_2(n) = 200000n^{1.5}$
   c) $f_3(n) = 1.00001^n$
   d) $f_4(n) = n^2$
   e) $f_5(n) = \binom{n}{2}$

2. Analyze the following algorithm and calculate the asymptotic complexity:

```
def process(numbers):
    # numbers is a list of numbers
    for i in range(len(numbers)-1):
        for j in range(len(numbers)-i-1):
            if numbers[j] > numbers[j+1]:
                numbers[j], numbers[j+1] = numbers[j+1], numbers[j]
            if j >= 2:
                break
    return numbers
```

3. Analyze and find the asymptotic(big-O) complexity of the following code written in python. Explain the analysis in full.

```
def process(arr, q, p, x):
    # array is a list of numbers
    if p >= q:
        y = (q + p)>>2  # >> symbol means right bit-shift
        if arr[y] == x:
            return y
        elif arr[y] > x:
            return process(arr, q, y - 1, x)
        elif:
            return process(arr, y + 1, p, x)
    elif:
        return -1
```

4. Draw recursion trees for the following code and try to find the asymptotic complexity for the following two functions (`function1` and `function2`):

```
def function1(array):
    # array is a python list of numbers
    length = len(array):
    if len == 1:
        return 0
    count = 0
    for i in range(length):
        for j in range(length):
            for k in range(length):
```

```
                for l in range(length):
                    count += 1
                    break
    mid = length // 2 # // in python means integer division
    array1 = array[0:mid] # assume this is a O(1) operation
    array2 = array[mid + 1:end] # assume this is a O(1) operation
    function1(array1)
    function1(array2)
    return count


def function2(array):
    # array is a python list of numbers
    length = len(array):
    if len == 1:
        return 0
    count = 0
    for i in range(length):
        for j in range(length):
            count += 1
    mid1 = length // 3 # // in python means integer division
    mid2 = 2*mid1
    array1 = array[0:mid1] # assume this is a O(1) operation
    array2 = array[mid1 + 1:mid2] # assume this is a O(1) operation
    array3 = array[mid2 + 1:end] # assume this is a O(1) operation
    function2(array1)
    return count
```

Are the asymptotic complexities for the two functions different? If so, justify the reason behind this change.