Data Structures and Algorithms

Workshop 2

1 Extra practice exercises

Exercise. How would you order the following functions in increasing order of complexity?

$$T_1(n) = 500n \log n^n + 899n$$
$$T_2(n) = (n+5)^3 + 8n^2$$
$$T_3(n) = \frac{2^n}{100}$$
$$T_4(n) = 9000n + 12189182$$

What is the big-Oh complexity in terms of the size of the input (eg a list length) of the following functions?

```
def f(L):
    """ assumes L is list of length n containing integers """
    u = 0
    for item in L:
        u = u + item**2
        u = u - item
    print(listSum)
```

```
def g(L):
    """ assumes L is list of length n """
    s = 0

```

```
    for index in range(1000):
        s = s + index

    for index in range(len(L)):
        L[index] = L[index] + 1
        for item in L:
            s = s + item
    print(L)
```

```
def F(n):
    """ assume n is a nonnegative integer """
    y = 0
    i = n
    while  i > 0:
        j = i
        while j > 0:
            y += 1
            j -= 1
        i -= 1
    return y
```

What does this function return for inputs $n = 28, m = 4$?

```
def F(n,m):
    """ assume n is a nonnegative integer """
    if m == 0: return 0
    return F(n-1,m-1)
```