Name (Student ID): Anik Patel (40091908) Teacher: Dhrubajyoti Goswami

Course-section: COMP 352-AA Submission Date: May 19th, 2019

Assignment 1 – Theory Questions

Q1

```
a) def generateRecursively(n):
       arr ← new int[n]
       for i, 0 to n-1:
           arr[i] ← i+1
       RecursiveP(arr, n)  → calling recursively
                             dominates the rest

   def RecursiveP(arr, n):
       if n is 1:         }
           print(arr)     }  constant
           return         }
       for i, 0 to n-1:  → O(n)
constant ← arr.swap(i, n-1)
           RecursiveP(arr, n-1)  → O(n)
constant ← arr.swap(i, n-1)

       → O(n) * O(n)   ⇒ O(n²)
```

b) def generateIterative(n):

```
def generateIterative(n):
    s1 ← new Stack(n)   // Stacks with length n
    s2 ← new Stack(n)
    for i, 0 to n:
        s1.push(i+1)
    iterativeP(s1, s2, n) → will dominate the rest

def iterativeP(s1, s2, n):
    for i, 1 to n!: → O(n!)
        temp ← i % n          } constant
        if i%2 = 0:
            while s1 is not Empty: → O(n)
                if temp=0:
                    val ← s1.pop()
                else:
                    temp ← temp-1          } constant
                    s2.push(s1.pop())
                s2.push(val)
                print(s2)
        else:
            while s2 is not Empty: → O(n)
                if temp=0:
                    val ← s2.pop()
                else:
                    temp ← temp-1          } constant
                    s1.push(s2.pop())
                s1.push(val)
                print(s1)
```

$(n!)(n)$

$\Rightarrow O(n(n!))$

Q2

a)

```
def  powerSet(T):
    start ← []  // Empty Set
    s ← new stack
    q ← new queue
    for i, all elements of T  // Puts all the numbers
        s.push(i)   from 1 to n into stack
    q.enqueue(start)
    while s is not Empty:
        val ← s.pop()  // element to add
                          this iteration
        while TRUE:
            begin ← q.dequeue()
            q.enqueue(begin + [val])  // concatinating starting set
                                         and value to add
            q.enqueue(begin)
            if begin is []:   // Keeps previously found set
                break
    return q
```

like a
do while

b) n is Nb of elements in T

```
def powerSet(T):
    start ← []                         } constant
    s ← new stack
    q ← new queue
    for i, all elements of T           } O(n)
        s.push(i)
    q.enqueue(start)                   } O(n)
    while s is not Empty:              } O(n)
        val ← s.pop()                  } constant
        while TRUE:
            begin ← q.dequeue()        } constant
            q.enqueue(begin + [val])   } O(n)
            q.enqueue(begin)
            if begin is []:            } constant
                break
    return q                           } constant
```

happens

$1 = 2^0$
$2 = 2^1$
$4 = 2^2$
$\vdots$
2^k

$$\underbrace{(1 + 2 + 4 + \dots + 2^k)}_{\text{geometric series}}$$

where $k = n$
↳ Nb of iterations for outer loop

$$2^{k+1} - 1$$

$$2^{n+1} - 2 \quad \boxed{\Rightarrow O(2^n)}$$

Q3

$n^4 + \log^2 n \Rightarrow O(n^4)$ 8

$\log \log n \Rightarrow O(\log \log n)$ 1

$\sqrt{n} \Rightarrow O(\sqrt{n})$ 2

$n! + n \Rightarrow O(n!)$ 9

$n/2 \Rightarrow O(n)$ 4

$\binom{n}{2} = \frac{n!}{2(n-2)!} = \frac{\cancel{1 \times 2 \times 3 \times \cdots \times n-2} \times n-1 \times n}{\cancel{1 \times 2 \times 3 \times \cdots n-2}} = \frac{(n-1)n}{2} \Rightarrow O(n^2)$ 6

$2^n \Rightarrow O(2^n)$ 7

$n \log n \Rightarrow O(n \log n)$ 3

$n^n \Rightarrow O(n^n)$ 10

$2^{\log n} = n \Rightarrow O(n)$ 5

$2^{n!} + n^2 \Rightarrow O(2^{n!})$ 12

$2^{2^n} \Rightarrow O(2^{2^n})$ 11

$$\boxed{\Rightarrow \log \log n \leq \sqrt{n} \leq n \log n \leq n/2 \leq 2^{\log n} \leq \binom{n}{2} \leq 2^n \leq n^4 + \log^2 n \leq n! + n \leq n^n \leq 2^{2^n} \leq 2^{n!} + n^2.}$$