Taylor Whitlock
CS 312 Section 002

1.7)

It takes O(m*n) time complexity. There will be m function calls because y will get halved each time, and though bit shifting left/right (adding/multiplying) as well as checking the last bit are all O(1) operations, addition is O(n) time since it has to go through each bit of n, thus the combination is O(m*n).

1.25)

```
def modexp(x, y, n):
    if y == 0:
        return 1
    z = modexp(x, y/2, n)
    return (z**2) % n if y % 2 == 0 else (((Z**2) % n)*x) % n

modexp(2, 125, 127)
    modexp(2, 62, 127)
        modexp(2, 31, 127)
            modexp(2, 15, 127)
                modexp(2, 7, 127)
                    modexp(2, 3, 127)
                        modexp(2, 1, 127)
                            modexp(2, 0, 127)
                                z = null, returns 1
                            z = 1, returns 2*(1**2) % n = 2
                        z = 2, returns 2*(2**2) % n = 8
                    z = 8, returns 2*(8**2) % n = 1
                z = 1, returns 2*(1**2) % n = 2
            z = 2, returns 2*(2**2) % n = 8
        z = 8, returns (8**2) % n = 64
    z = 64, returns 2*(64**2) % n = 64


2**125 (mod 127) = 64
```

Problem 3)

x	y	z	n	return value	call level
2	21	16	18	8	1
2	10	14	18	16	2
2	5	4	18	14	3
2	2	2	18	4	4
2	1	1	18	2	5
2	0	None	18	1	6