HW #2 - 1.7, 1.25, 2^{21} mod 18

1.7) How long does the recursive multiplication algorithm take to multiply an n-bit number by an m bit number? Justify.

```
function multiply (x, y)
  if y=0 : return 0
  z = multiply (x, [y/2])
  if y is even
      return 2z ] O(n)
  else
      return x+2z ] O(n)
```

m recursive calls

$\overset{101}{x=5}$ (3 bits) $\overset{1011}{y=10}$ (4 bits)

x	y	z	return
5	10	25	50
5	5	10	25
5	2	5	10
5	1	0	5
5	0		0

The way the algorithm works, each y/2 recursive call removes 1 bit from m, so we will have m recursive calls. During each recursive call we make either 1 multiplication (if y is even) or 1 add & 1 multiplication, both of which can be considered as O(n) linear time. With that we could say our function runs with O(n·m) complexity with m being the # of bits/recursive calls made.

1.25) Calculate $2^{125} \bmod 127$ using any method.

$a = 2 \quad N = 127$

Since 127 is prime, by Fermats little theorem $a^{N-1} \equiv 1 \bmod N$

$2^{126} \equiv 1 \bmod 127$ which can be broken down to

$2 \cdot 2^{125} \equiv 1 \bmod 127$ $\quad x = 2^{125}$

$2x \bmod 127 \equiv 1$ In order to solve $2x$ will need to $= 128$

$\frac{2x}{2} = \frac{128}{2}$

$x = 64 \rightarrow$ Therefore we know $2^{125} = 64$

$2 \cdot 64 \equiv 1 \bmod 127$ ✓

3) Solve $2^{21} \bmod 18$ using 1.4 formula. Show work on table

```
modexp(x,y,N)
  if y=0: return 1
  z=modexp(x,[y/2],N)
 If y is even
   return z^2 mod N
 else
   return x z^2 mod N
```

x	y	Yold	pwr of x	z	return
2	21	1	x	16	512 mod 18 = 8
2	10	0	x^1	14	196 mod 18 = 16
2	5	1	x^4	4	32 mod 18 = 14
2	2	0	x^8	2	4 mod 18 = 4
2	1	1	x^{16}	1	2 mod 18 = 2
2	0				

$\boxed{2^{21} \bmod 18 = 8}$