Homework #2 Key

1.7

Assume we want to multiply the n-bit number x by the m-bit number y. The algorithm must terminate after m recursive calls, because at each call y is halved (the number of digits is decreased by one). Each recursive call requires a division by 2, a check for even-odd, a multiplication by 2 (all of those take constant time) and a possible addition of x to the current result (which takes $O(n)$ time). Thus, the total is $O(m \cdot n)$ time.

1.25

Since 127 is prime, by Fermat's Little Theorem, $2^{126} \equiv 1 \pmod{127}$. Then $2 \cdot 2^{125} \equiv 1 \pmod{127}$ and we are looking for a number that, when multiplied by two, is congruent to (or has a remainder of) $1 \pmod{127}$. Since $2 \cdot 64 \equiv 1 \pmod{127}$ and $2 \cdot 2^{125} \equiv 1 \pmod{127}$, then $2^{125} \equiv 64 \pmod{127}$. So the answer is 64.

Alternatively, $2^{125} \equiv 2^{119} \cdot 2^6 \equiv (2^7)^{17} \cdot 2^6 \equiv 1^{17} \cdot 2^6 \equiv 2^6 \equiv 64 \pmod{127}$.

Modexp algorithm for $2^{21} (\mathbf{mod} 18)$

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

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