J.B. Morris

0.1

a) $f = \Theta(g)$
b) $f = O(g)$
c) $f = \Theta(g)$
d) $f = O(g)$
e) $f = \Theta(g)$
f) $f = \Theta(g)$
g) $f = O(g)$
h) $f = \Omega(g)$
i) $f = O(g)$
j) $f = O(g)$
k) $f = \Omega(g)$
l) $f = O(g)$
m) $f = O(g)$
n) $f = \Theta(g)$
o) $f = \Omega(g)$
p) $f = O(g)$
q) $f = \Theta(g)$

0.4

a) $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} e & f \\ g & h \end{pmatrix}$
   $(a \cdot e) + (b \cdot g)$
   $(c \cdot e) + (d \cdot g)$
   $(a \cdot f) + (b + h)$
   $(c \cdot f) + (d \cdot h)$

b) Ex: $x^{400} = x^{256} \cdot x^{128} \cdot x^{16}$
   Find the highest power of 2 that can be subtracted from the power and subtract the next highest power from the difference and repeat the process until the sum of the powers of 2 adds up to the total exponent.

1.2 $\log(n!) = \Theta(n \log(n))$

Upper: $n! = n^n$
$\log(n^n) = n \log(n)$
Lower: $n! = (n/2)^{n/2}$
$\log((n/2)^{n/2}) = n/2(\log(n/2)) = n/2 \log(n) – \log(2) = n \log(n)$

1.11 yes

1.13 yes

1.16 $a^{57} = a^{32} + a^{16} + a^{8} + a$

1.25 $2^{125} \bmod 127 = 64$

1.33

```
def lcm(x, y):
    return (x*y)/gcd(x, y)


def gcd(x, y):
    while y:
        x, y = y, x % y
    return x

input1 = int(input("Enter number: "))
input2 = int(input("Enter second number: "))
print("The least common multiple of", input1, " and ", input2, " is ", lcm(input1, input2))
```

complexity: $\Theta\ (n^3)$

1.35 There is no efficient way to calculate the factorial for a large number (N – 1)

1.39

```
def modpow2(base, exponent, exponent2, mod):
    c = 1
    for i in range(0, exponent*exponent2):
        c = (c * base) % mod
    return c

input1 = int(input("Enter base: "))
input2 = int(input("Enter exponent: "))
input3 = int(input("Enter exponent's exponent: "))
input4 = int(input("Enter mod: "))
print("modpow = ", modpow2(input1, input2, input3, input4))
```