Jacob Rammer

Assignment 1

1. $313n^3 = 2n^4$ $\qquad$ $313(156.5)^3 \leq 2(156.5)^4$
$= 313 = 2n$ $\qquad$ $1199740620.13 \leq 1199740620.13$ ✓
$= n = \frac{313}{2} = 156.5$

2. The def of Θ: $0 \leq c_1 n^b \leq (n+b)^b \leq c_2 n^b \;\; \forall\, n \geq n_0$
$N = 2|A|$
Since $a \leq n$, we know that $a+n \leq 2n$. If we half $n \rightarrow \frac{1}{n}n$, it is also clear that $a+n \geq \frac{1}{2}n$ when $|a| \leq n$ when $|a| \leq \frac{n}{2}$. So when $N \geq 2|A|$ this means $0 \leq \frac{n}{2} \leq n+a \leq 2n$
With some algebra, I know I can raise this to a power of $b \in \mathbb{Z}^+$ $(b>0)$. Now I have: $0^b \leq \left(\frac{n}{2}\right)^b \leq (n+a)^b \leq (2n)^b$
Simplifying: $0 \leq \frac{1}{2^b}(n^b) \leq (n+a)^b \leq 2^b \cdot n^b$

Where $c_1 = \frac{1}{2^b}$, $c_2 = 2^b$, $N = 2|A|$ ✓

3.

	A	B	O	o	Ω	ω	Θ
a	$\lg^k n$	n^{ϵ}	Y	Y	N	N	N
b	n^k	c^n	Y	Y	N	N	N
c	$\sqrt{n}$	$n^{\sin(n)}$	N	N	N	N	N
d	2^n	$2^{n/2}$	N	N	Y	Y	Y
e	$2^{\lg c}$	$c^{\lg n}$	Y	N	Y	N	Y
f	$\lg(n!)$	$\lg n^n$	Y	N	Y	N	Y

4. 1 hr $= 3600000$ (n var calculator)
a) $\frac{X}{50} = \frac{36{,}00000}{.4} = 4500000000$

b) $\frac{X \log X}{50 \log 50} = \frac{36.00000}{36.00000} = 299618332$
$.4$

c. $\frac{x^3}{50^3} = \frac{3600000}{.4} = 10400$

d. $\frac{2^x}{2^{50}} = \frac{36,000000}{.4} = 73$

5. a) $\Theta(n)$

b) Since the summation is to n:
we have:

```
y = 0
for i in range(n)
    a = 1
    for j in range(0, i)
        a *= x  (x is from summation)
    y += a
```

c) $y = \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k$

Initialization: $y = 0$, $i = n$

$= y = a_i + x \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k = a_i + x \sum_{k=1}^{n-i} a_{k+i} x^{k-1} = \sum_{k=0}^{n-i} a_{k+i} x^k$

maintance: $y = a_i + x \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k = a_i + \sum_{k=0}^{n-i-1} a_{k+i+1} x^{k+1}$

$= a_i \sum_{k-1}^{n-i} a_{k+i} x^k$

Termination: (for $i = n$ downto 0) - that means this loop will terminate when $i = -1$.

using $y = \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k$ and subsituting i, we get

$$\sum_{k=0}^{n-(-1+1)} a_{k+i+1} x^k = \sum_{k=0}^{n} a_k x^k \quad \square$$

d) By proving the loop invariant above, upon termination the algorithm calculates correctly

6. What we want to do is keep pushing / enqueueing until pop / dequeue is called, which will then dequeue everything into the first stack into the second.

```
Stack1 = Stack()
Stack2 = Stack()

def enqueue(x):
    Stack1.push(x)

def dequeue(Stack)
    if len(Stack2) != 0:
        while len(Stack1) != 0:
            Stack2.push(Stack1.pop())
    return S2.pop()
```

runtime: enqueue: $O(1)$

dequeue: $O(n)$ - because we have to move every element from Stack1 to Stack 2.