Tristan Rice (q7w9a, 25886145)
19/10/2015
CPSC221 - Written Assignment 1

Question 1
This code will never delete the head node since it starts operating on head->next. You could add something like:

```
if (head->value == x) {
   head = head->next
}
```

Question 2
a) If count(pop) != count(push) the sequence is invalid.
b) <3,2,5,6,4,1> = IIIOOIIOIOOO
<1,5,4,6> = IOIIIIOOIO
At this point the stack is [2,3]. Thus, there's no way to use pop to get <2,3> by removing elements from the back.

c) Once a bigger number is visited, smaller numbers can only be outputted in reverse order.
Proof.
Assume that it is possible to output π if for some numbers i < j < k, $\pi(j) < \pi(k) < \pi(i)$.
Let $\pi(i) = 3, \pi(k) \ = \ 2, \pi(j) = 1$
$\pi = \ < 3{,}1{,}2 >$
Push 1, 2, 3
Stack is now <1,2,3>
Pop 3
Stack is now <1,2>
The next pop needs to return 1. However, pop will return 2 first.
Thus, by contradiction, it is impossible to output π if for some numbers i < j < k, $\pi(j) < \pi(k) < \pi(i)$.

d) No, you can only output the numbers in order despite order of pops and pushes.

Question 3
a) $\lg 8n = \lg(23)n = \lg 23n = 3n$
b) $2\lg(nm) - \lg(m2) = 2\lg(nm)2\lg(m2) = nmm2 = nm$
c) $-\lg 164 = \lg 64 = 6$
d) $\log p1p = \log pp{-}1 = -1$
e) $8\lg n = (23)\lg n = (2\lg n)3 = n3$

Question 4
$$\sqrt{n} \le n \le (\log n)^4 \le n^{2\log n} \le \sum_i^n i^2 \le 2^n \le 2^{n^2} \le 2^{2^n} \le n^{2\lg n} \le \log(n!)$$

Question 5
b) $T(n) = (n+4)(6n+7), \Theta(1)$
c) $T(n) = \sum_{i=0}^{n} 2i + 3, \Theta(n+1)$
d) $\Theta(n+1)$
e) $\Theta(\lg(n+1))$
f) $T(n) = \sum_{i=1}^{n} \sum_{j=1}^{i} 3i^3$

$3i3,\Theta(1)$

$\sum_{j=1}^{i} 3i^3, \Theta(i)$

This reduces to

$\Theta(\sum_{i=1}^{n} i)$

$\Theta(\frac{n(n+1)}{2})$

Proof by induction.
Base case:

$T(1), \Theta(\frac{(1+1)(1)}{2}) = \Theta(1)$

Inductive step

$\Theta(n+1) = \Theta(n) + (n+1)$

$= \frac{(n+1)((n+1)+1)}{2} = \frac{(n+1)n}{2} + (n+1)$

$(n+1)(n+2) = n^2 + n + 2(n+1)$

$n^2 + 3n + 2 = n^2 + n + 2(n+1)$

$2n + 2 = 2(n+1)$

Identity. Thus the runtime of T(n) is $\Theta(n(n+1)2)$

Question 6

a)

```
Pow(x, n)
    result = x // T(n) = 1
    count = 1 // T(n) = 1
    while 2*count <= n { // T(n) = 1 + floor(lg(n))
        ... // floor(log(n))
    }
    while count < n { // 1 + n - 2^floor(lg(n))
        ... // n - 2^floor(lg(n))
    }
T(n) = 4 + 3*floor(lg(n)) + 3*(n-2^floor(lg(n)))
```

Thus, $\Theta\left(\lg(n) + n - 2^{\lg(n)}\right)$

b)

```
longestIncreasing(A)
    ResultForPrefix = new Array[A.length] // T(n) = 1
    for i = 0 to A.length - 1 { // T(n) = n
        r = 1 // T(n) = n
        for j = 0 to i - 1 { // T(n) = sum i=0 to n: i
            ... // T(n) = sum i=0 to n: i
         }
         ResultForPrefix[i] = r
         if bestOverall < r then bestOverall = r
     }
     return bestOverall // T(n) = 1
T(n) = 2 + 4*n + 3*n(n+1)/2
```

$\Theta\left(n^2\right)$

Question 7

```
heapify(heap, size) {
    for(i = (size - 2) / 2; i >= 0; i--)
        swapDown(heap, i, size);
}
```

Start: E C D A G B F I H

```
E C D A G B F I H
    F
  D
    B
E
    G
  C
      H
    A
      I

E C B A G D F I H
    F
  B
    D
E
    G
  C
      H
    A
      I

E A B C G D F I H
    F
  B
    D
E
    G
  A
      H
    C
      I

A C B E G D F I H
    F
  B
    D
A
    G
  C
      H
    E
      I
```

Question 8

```
void printMinimum(heap, i, size, q) {
     if (i < size && heap[i] < q) {
         print(heap[i])
         printMinimum(heap, 2 * i + 1, size, q)
         printMinimum(heap, 2 * i + 2, size, q)
    }
}
```