CS 221 Assignment 1

Anthony Chui
Adam Yin

1) The code only handles cases where there are values to be deleted that are not the head. Also it does not free deleted nodes and will cause memory leaks. Add these lines of code:

```
if (head == NULL) return; //Check null head
while(head.value == x){ //Delete all head of value x
    node = head;
    head = head.next
    delete node;
}
if (head == NULL) return; //Check null head again
```

Then proceed with the rest of the given code, ensuring ~~deleted~~ nodes ~~with~~ are deleted as well.

2) a) - Throughout a sequence, there must be more or equal push than pop operations
- The end of a sequence must mean the same amount of push and pop operations.

b) 1, 5, 4, 6, 2, 3 requires IOIIIIOOIO... and stops before outputting 2 because 3 needs to be popped before 2. The sequence is impossible.

→

2) C) Assume $i < j < k$ and that it is possible that $\pi(j) < \pi(k) < \pi(i)$ in a permutation π.

In order to reach the i^{th} index of the sequence, there must be i pops (and thus at least i pushes). Because $\pi(i) > \pi(j)$, there must be more net pops than pushes from index i to j (otherwise a greater number at j will be popped). However, $\pi(k) > \pi(j)$ and so there must be more net pushes than pops from j to k. However, more pushes than pops will cause $\pi(k)$ to be greater than $\pi(i)$ since only numbers greater than $\pi(i)$ can be pushed and popped now. This contradiction means the initial conditions are impossible.

D) A queue would only generate one possible permutation because the order that numbers are queued are the order they are dequeued. The order of queue/dequeues don't matter because the oldest element will always be dequeued first, so the only sequence is $1, 2, 3 \ldots n$.

3) a) $\lg 8^n = n \lg 8 = 3n$

b) $2^{\lg(nm) - \lg(m^2)} = 2^{\lg \frac{n}{m}} = \frac{n}{m}$

c) $-\lg \frac{1}{64} = \lg 64 = 6$

D) $\log_p\left(\frac{1}{p}\right) = -\log_p(p) = -1$

E) $8^{\lg n} = 2^{3 \lg n} = n^3$

4) $\sqrt{n}, (\log n)^4, n, \lg(n!), \sum_{i=1}^{n} i^2, n^2 \log n, n^2 \lg n, 2^n, 2^{n^2}, 2^{2^n}$

5) b) $\Theta(n^2)$

c) $\Theta(2^n)$

d) $\Theta(2^n)$

e) $\Theta(\lg(n))$

f) $\sum_{i=1}^{n} \sum_{j=1}^{i} 3i^3 = \sum_{i=1}^{n} 3i^3$

Summation formula $= 3\left(\frac{n^2(n+1)^2}{4}\right) = \frac{3}{4}(n^4 + 2n^3 + n^2)$

$an^4 \leq \frac{3}{4}n^4 + 2n^3 + n^2 \leq bn^4$

where for an^4, $a = \frac{3}{4}$, $n \geq 0$

for bn^4, $b = 1$, $n \geq 3 + 2\sqrt{3}$

thus $\frac{3}{4}(n^4 + 2n^3 + n^2) \in \Theta(n^4)$

6) a) The worst case exists when adding 1 to n and lg applied it would cause the output to not have a decimal. The first while loop runs for $2\lg n$ iterations, due to the fact that count is successively multiplied by 2 and there are 2 lines of code. The second loop runs for $2(n - 2^{\text{floor}(\lg(n))})$ iterations since it →

has to take the remainder of n and 2 to the whole power of $\lg(n)$ since the first while loop converts it as such. Also, there are 3 individual lines of code in the function so

$T(n) = 2(n - 2^{floor(\lg(n))}) + 2\lg(n) + 3$

$2^{floor(\lg(n))}$ is asymptotically similar to n which are both asymptotically greater than $\lg(n)$ thus $\theta(T(n)) \in \theta(n)$

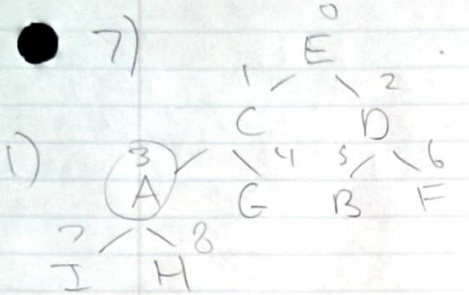
b) The worst case exists when every if condition is true for the inner loop. Since it iterates from $j = 0$ to the outer loop index, it will iterate for $0, 1, 2 \ldots i-1$ which can be represented by the summation formula $\frac{A(A+1)}{2} - A$ since it is offsetted by 1 index. Thus, each iteration of the outer loop will result in $\frac{A(A+1)}{2} - A + (A)(3)$, Add 1 for the return statement

So $T(n) = \frac{A^2}{2} + 2.5A + 1$

where $aA^2 \leq \frac{T(n)}{2} \leq bA^2$

where $a = 0.5$ for $A \geq 0$ and
$b = 1$ for $A \geq 5.4$

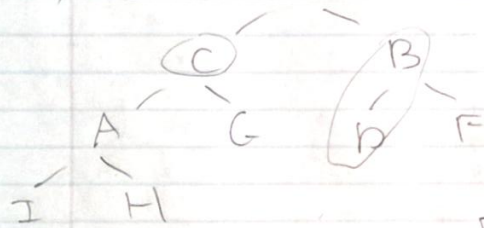
7) E C D A G B F I H



$\frac{\text{length}=9}{2} - 1 = 3$

No swap needed, check index 2

2) E C B A G D F I H

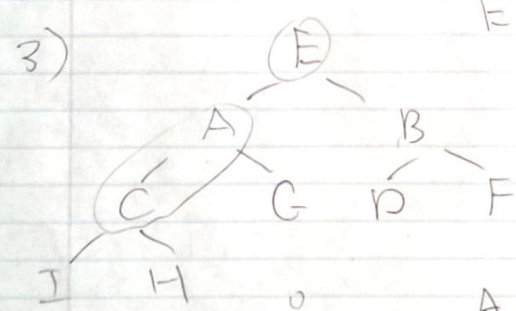


Swap B with D.
Check index 1

E A B C G D F I H

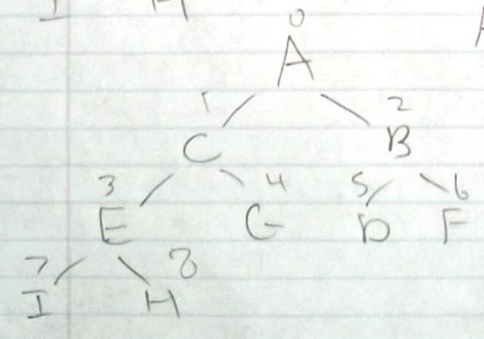
Swap C with A.
Check index 0

3)



A C B E G D F I H

Swap E down to C, swapping with A and C in the process.



8) Start at root node and check if it is less or equal to q. If so, output its value. Recursively repeat that process for the left and right node.