CSE 332: Data Structures and Parallelism

Section 3: BSTs, Recurrences, and Amortized Analysis

0. Interview Question: Binary Search Trees
Write pseudo-code to perform an in-order traversal in a binary search tree without using recursion.

1. Recurrences and Closed Forms
For the following code snippet, find a recurrence for the worst case runtime of the function, and then find a closed form for the recurrence.

Consider the function $f$:

```plaintext
1  f(n) { 
2    if (n == 0) {
3      return 1;
4    }
5    return 2 * f(n - 1) + 1;
6  }
```

- Find a recurrence for $f(n)$.

- Find a closed form for $f(n)$.
2. Recurrences and Closed Forms
For the following code snippet, find a recurrence for the worst case runtime of the function, and then find a closed form for the recurrence.

Consider the function $g$:

```java
1 g(n) {
2    if (n < 3) {
3        return 1000;
4    }
5    if (g(n/3) > 5) {
6        for (int i = 0; i < n; i++) {
7            System.out.println("Yay!");
8        }
9        return 5 * g(n/3);
10    } else {
11        for (int i = 0; i < n * n; i++) {
12            System.out.println("Yay!");
13        }
14        return 4 * g(n/3);
15    }
16 }
```

- Find a recurrence for $g(n)$.
- Find a closed form for $g(n)$.

3. MULTI-pop
Consider augmenting the Stack ADT with an extra operation:

`multipop(k)`: Pops up to $k$ elements from the Stack and returns the number of elements it popped

What is the amortized cost of a series of push’s, Stack assuming push and pop are both $O(1)$?