CMSC132 Fall 2018 Trees, Big O Worksheet

1. For each of the following code fragments, circle the critical section, determine T(n) and the complexity (Big O).

   a.

```
for (int i = 10; i <= n - 2; i++) {
       for (int k = 1; k <= n; k *= 2) {
              sum = i + k;
       }
}
```

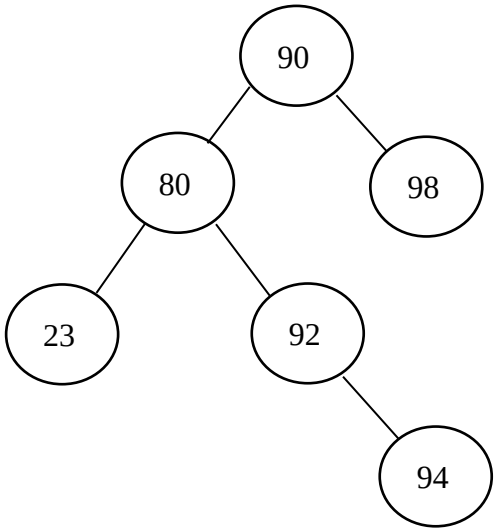
   b.

```
for (int i = 1; i <= n / 2; i += n/2) {
       for (int k = 1; k <= 100; k++) {
              System.out.println(i * k);
       }
}
```

2. When do we want to use a recurrence relation?

3. What is Big-Omega?

4. For the following tree, perform a preorder, and inorder and a postorder traversal.



5. Is the previous tree a Binary Search Tree? If not, update the tree so it becomes one.

6. What is the best way to insert values into a tree so the tree is balanced? Does adding values to the tree in sorted order helps?

7. When we insert a node into a binary search tree, does it always become a leaf?

8. Insert into a binary search tree the same value twice, using the right subtree to handle duplicates.