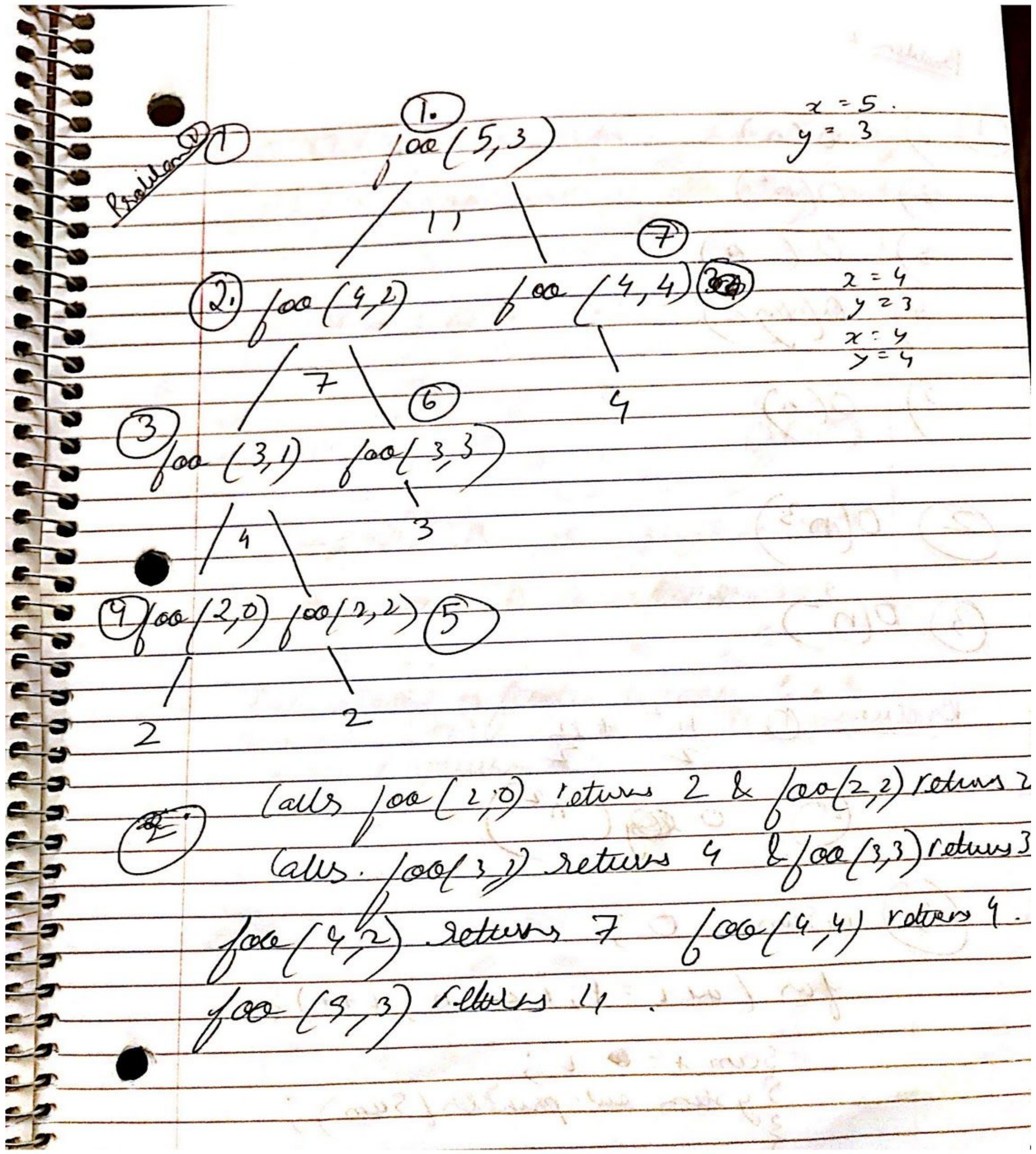
Problem Set 5, Part I

Problem 1: A method that makes multiple recursive calls
1-1)



2.
Calls foo(2,0) returns 2 & foo(2,2) returns 2
Calls. foo(3,1) returns 4 & foo(3,3) returns 3
foo(4,2) returns 7 foo(4,4) returns 4.
foo(5,3) returns 11 .

Scanned with CamScanner

call 4 returns 2
call 5 returns 2
call 3 returns 4
call 6 returns 3
call 2 returns 7
call 7 returns 4
call 1 returns 11

Problem 2: Computing Big-O

2-1) a(n) = O(n)
b(n) = O(n^2)
c(n) = O(n)
d(n) = O(log(n))
e(n) = O(n)

2-2) O(n^3)

2-3) O(nlog(n))

Problem 3: Sum generator

3-1) (n^2 + n)/2

3-2) O(n^2) because the highest order of n in the formula for the sum is n^2

3-3)
```
public static void generateSums(int n){
        int sum=0;
        for(int i=1; i<=n; i++){
                sum+=i;
                System.out.println(sum);
        }
}
```

3-4) O(n) because there is only one for loop that iterating n times.

Problem 4: Comparing two algorithms

4-1) Best case: O(n^2)
Average case: O(n^2)
Worst case: O(n^2)
Algorithm A is selection sort and has an average time efficiency of O(n^2)

4-2) Best case: O(n)
Average case: O(n^2)
Worst case: O(n^2)
Algorithm B is bubble sort and has an average time efficiency of O(n^2)

4-3) Algorithm B is more efficient because its time efficiency is O(n)
in the best case while Algorithm A is always O(n^2).