```java
/**
 * This is my program for figuring out the average runtime of A(n)
 *
 * Throughout the code is comments on the reason I choose to make certain decisions
 * My analysis of the results is on the next page
 */
class test {
    Run | Debug
    public static void main(String[] args) {
        for (int n = 5; n <= 100; n += 5) {
            long count = 0; //This is the variable I use to derive the number of calls the program makes
            for (int h = 1; h <= 10; h++) {
                int m = (int) (Math.random() * n + 1); //This is the my method of assigning a randown integer
                count += r(n, m); //Represents the numerator asked for in part 2
            }
            count /= 10; // Represents the denominator asked for in part 2

            System.out.println(n + ", " + count + ", " + count / Math.pow(2, n));
            //Above is the print line asked for in part 3
        }

    }

    public static long RecBin(int n, int m) {
        //This given function calculates the amount of endpoints in the recursive tree
        if (m == 0 || m == n){
            return 1;
        }
        //Decided to make these variables long because after n=30 there is an interger overflow
        long a = RecBin(n - 1, m - 1);
        long b = RecBin(n - 1, m);
        return a + b;
    }

    public static long r(int n, int m) {
        //This fabricated function calculates the amount of calls in the recursive tree
        if ( m == 0 || m == n){
            return 1;
        }
        //Decided to make these variables long because after n=30 there is an interger overflow
        long a = r(n - 1, m - 1);
        long b = r(n - 1, m);
        return a + b + 1; //This is the key different adding +1 to every return here gets the correct number
    }
}
```

```
n   e(n)          e(n)/2^n

5,  11,           0.34375
10, 210,          0.205078125
15, 4077,         0.124420166015625
20, 212730,       0.20287513732910156
25, 1843684,      0.054946064949035645
30, 74274399,     0.06917342450469732
35, 622317486,    0.01811182260280475
40, 94030025938, 0.08551981039818202
```

When comparing e(n) to the "best case" (constant) and the "worst case" (2^n) the program does not end any clean way. It lands on an "inbetween" amount. The average for e(n)/2^n is about 0.138 which clearly places it way above the n case and way below the 2^n case. Another quirk that is identified with this data is that generally, as n increases the e(n)/2^n decreases. It is unkown whether this trend continues or fades off logorithmically. Unfortunately, this program is seemingly either inefficient or the numbers trying to be calculated are throttled because whenever my device tried to calculate above n = 30 the time needed to do each level was noticebly exponential. It toke about 15 mins for my computer to get to the n = 40 case after the n = 35 case. I choose to cut the data there because going any further would have taken an extraordinarily long time.