0.1. Problem 1. Running Time Analysis (18 points)

Provide tight asymptotic bounds (i.e. $\Theta(\cdot)$) on the worst case running times of the following two procedures. Please provide a very brief explanation of your answer. (7 points each)

(a)

```
procedure BackForwardAlg(n)
    if n≤10 then
        return n
    if n even then
        return BackForwardAlg(n/2)
    else
        return BackForwardAlg(n+ 3)
```

This algorithm is $\Theta(\log n)$. We have the base case which is $O(1)$ and we have two recursive calls, only one of which executes. If it is an even number, the number is halved, if not 3 is added to it, which makes n and even number, so at most it makes log n recursive calls on half, or adds 3 on log n recursive calls which is equal to $2 \log n = O(\log n)$.

(b)

```
procedure RecursiveAlg(A[1 . . . n])
    if n== 1 then
        return False
    mid =⌈n/2⌉

    for i= 1 to mid do
        for j=mid + 1 to n do
            if A[i] == A[j]then
                return True

    return (RecursiveAlg(A[1 . . . mid]) || RecursiveAlg(A[mid + 1 . . . n]))
```

We have to base case $T(1) = O(1)$. There is $\frac{1}{4}n^2$ work inside of the loop or $\Theta(n^2)$ our recurrence then makes two recursive calls on half of the items, we get $T(n) = 2T\left(\frac{n}{2}\right) + n^2$ which is $\Theta(n^2)$.

(c) (4 points) Give a one sentence description of what RecursiveAlg does.
This algorithm returns true if there are any duplicate values in the array.

0.1. Problem 2. Asymptotic Bounds (20 points)

Provide tight asymptotic bounds (i.e. $\Theta(\cdot)$) on the following. Correct answers are given full points without explanation. (5 points each)

(a) $\log^3 n + 5^{\log\log n} + \log^2(\log^4 n)$
This function is $\Theta(5^{\log\log n})$

(b) The number of bits needed to write 12^n binary.
$\log(12^{n)} = n\log(12) = \Theta(n)$

(c) Planet X has a current population of 1 billion. Assuming that the population of planet X doubles every year, give an asymptotic bound on the population of planet X in 100 years from now.
$\Theta(2^n)$

(d) $\sum_{i=1}^{n} \log \frac{n}{i}$ [Hint: Stirling's approximation may be useful (see Wikipedia).]

0.1. Problem 3. Ordering functions (25 points)

Sort the following 25 functions from asymptotically smallest to asymptotically largest, indicating ties if there are any. You do not need to turn in proofs (in fact, please don't turn in proofs).
$\cos(n)+2,\ \log^*(\log n),\ \log^*(2^{2^{2^n}}),\ \log(\sqrt{\log n}),\ 12+\log(\log(n)),\ \sqrt{\log n},\ \lfloor \log n \rfloor,\ \sum_{i=1}^{n} \frac{1}{i},\ \log^{201} n,\ (\log^* n)^{\log n}$
$,(\log n)^{\log n},\ n^{\frac{1}{125}},\ n,\ n\log^4 n,\ \left(\frac{n}{2}-1\right)^{2.1},\ \sqrt{n}^e,\ \sum_{i=1}^{n} i^2,\ n^3,\ 2^{\log n},\ 4^{\log n},\ n^{\frac{1}{\log\log n}},\ n^{\frac{3}{2\log n}},\ n^{\log\log n},\ \left(1+\frac{1}{250}\right)^n,\ n^{\sqrt{n}}$

0.1. Problem 4. Writing Recurrences (12 points)

6 points each. For both problems, don't forget to state the base case(s). You do not need to solve your recurrences.

(a) For some positive integer n, consider a 2×n rectangle. Suppose you wish to cover this rectangle with n smaller 1×2 rectangles, each of which can either be placed horizontally (i.e. 1 × 2) or rotated and placed vertically (i.e. 2 × 1). Write a recurrence describing the number of possible ways to cover the 2 ×n rectangle with these smaller rectangles.
$T(1) = 1\ T(n) = T(n-1) + 2T\left(\frac{n}{2}\right)$

(b) You have an unlimited supply of red, blue, and green plates in your kitchen. Red and blue plates have thickness 1 and green plates have thickness 3, and you are going to create a stack of total thickness exactly n. Write a recurrence counting the number of different possible color orderings for the stack of thickness n.
$T(0) = 1, T(1) = 2\ T(n) = 2T(n-1) + T(n-3)$

0.1. Problem 5. Bounding Recurrences (27 points)

Provide tight asymptotic bounds for the following six recurrences (i.e. $\Theta(\cdot)$). If you cannot give matching upper and lower bounds, give the best upper and lower bounds you can. For the first three you must specify what the level sum (i.e. total cost) is of ith level in the recursion tree. For the last three you do not need to specify the level sum, but you should give a very brief explanation.

(a)$T(n) = 6T(n/3) + n^2$
$i_0 = n^2$
$i_1 = 6(\frac{n}{3})^2$
$i_2 = 36(\frac{n}{9})^2$
$i_i = \frac{6^i n^2}{3^{2i}}$
This recurrence is $\Theta(n^2)$.

(b) $T(n) = 7T(n/3) + 4n^{3/2}$
$i_0 = 4n^{\frac{3}{2}}$
$i_1 = 28\left(\frac{n}{3}\right)^{3/2}$
$i_i = 28^i\left(\frac{n}{3^i}\right)^{3/2}$
This recurrence is $\Theta(n^{\log_3 7})$

(c) $T(n) = 8T(n/2) + n^3$
$i_0 = n^3$
$i_1 = n^3$

$i_i = n^3$
This recurrence is $\Theta(n^3 \log n)$

(d) $T(n) = T(n/7) + T(n/11) + \sqrt{n}$
$i_0 = \sqrt{n}$
$i_1 = \sqrt{\frac{n}{7}} + \sqrt{\frac{n}{11}}$

This recurrence is $\Theta(\sqrt{n})$ each level is going down by a constant factor.

(e) $T(n) = T(\sqrt{n}) + 3$
This recurrence is $\log^* n$, as we end up with the reiterated square root of n with a constant time driving function.

(f) $T(n) = T(\log n) + \log n$
Our recursion tree is just a straight line of nodes, at each level we are reapplying the logarithm to n and doing log work. The total at each level is decreasing at a very vast rate, so the root node is the largest sum by a lot. This recurrence is $\Theta(\log n)$.