CS 361, HW4

Prof. Jared Saia, University of New Mexico

Due: Sept. 29th

1. Consider the recurrence $T(n) = 2T(n/4) + n^2$
   (a) Use the recursion tree method to get a tight upper bound (i.e. big-O) on the solution to this recurrence
   (b) Now use annihilators (and a transformation) to get a tight upper bound on the solution to this recurrence. Show your work. (Note that your two bounds should match)

2. Consider the recurrence $T(n) = 2T(n/2) + \log^2 n$
   (a) Use the Master method to get a general solution to this recurrence.
   (b) Now use annihilators (and a transformation) to get a tight upper bound on the solution to this recurrence. Show your work. (Note that your two bounds should match)

3. Consider the following function:

```
int f (int n){
  if (n==0) return 0;
  else if (n==1) return 1;
  else{
    int val = 6*f (n-1);
    val = val - 9*f (n-2);
    return val;
  }
}
```

   (a) Write a recurrence relation for the *value* returned by f. Solve the recurrence exactly. (Don't forget to check it)

(b) Write a recurrence relation for the *running time* of f. Get a tight upperbound (i.e. big-O) on the solution to this recurrence.

4. Consider the following function:

```
int f (int n){
  if (n==0) return 0;
  else if (n==1) return 1;
  else{
    int val = 4*f (n-1);
    val = val - 4*f (n-2);
    return val;
  }
}
```

(a) Write a recurrence relation for the *value* returned by f. Solve the recurrence exactly. (Don't forget to check it)

(b) Write a recurrence relation for the *running time* of f. Get a tight upperbound (i.e. big-O) on the solution to this recurrence.