Analysis of Algorithms Homework

Answer questions 1-4 and submit your solution into the dropbox by the required due date.

1. For each pair of functions (f(n) and g(n)) in the table below, enter true or false to indicate whether or not f is $O, \theta, or\ \Omega$ of g on input of size n. Assume $k \geq 1$ and $c > 1$ are constants.

$f(n)$	$g(n)$	$f(n) = O(g(n))$	$f(n) = \theta(g(n))$	$f(n) = \Omega(g(n))$
$n^2 + 3n$	$5n^2 + 3n + 8$			
3^n	2^n			
n^k	c^n			
c^k	n			

2. Give an expression for the running time (T(n)) each of the following fragments of code and give an asymptotically tight bound for $\theta(g(n))$ (Big Theta) on your solution.

```
// code fragment A
for ( i = 1; i <= N; i++ )
{
   for ( j = 1; j <= i; j++ )
   {
      for ( k = 1; k <= 10; k++ )
      {
         count++;
      }
   }
}
```

```
// code fragment B
for ( i = 1; i <= n; i++ )
{
   for ( j = 1; j <= n; j++ )
   {
      for ( k = 1; k <= j; k++ )
      {
         count++;
      }
   }
}
```

```
// code fragment C
for ( i = 1; i <= n; i++ )
{
   if ( i % 2 == 0 )
   {
      for ( j = 1; j <= n; j++ )
      {
         count++;
      }
   }
   else
   {
      for ( k = 1; k <= j; k++ )
      {
         count++;
      }
   }
}
```

```
// code fragment (recursive function) D
void fun( int p1, int p2 )
{
   if ( p1 <= p2 )
   {
      int mid = split( p1, p2 );
      fun( p1,  mid );
      fun( mid, p2 );
   }
}
```

Assume the function fun is invoked with the initial values of p1=1 and p2= N. Also assume the time needed to execute the split function is N (N=p2-p1+1). The split function divides the array into two equal parts of size N/2 (ignore the fact that sometimes the split would produce intervals differing in size by 1). You may ignore the cost of the conditional statement and the assignment in your recurrence relation.

3. Given two correct solutions to the same problem with the following asymptotically tight bounds on time complexity:

Program 1: $T(n) = \theta(n^2)$

Program 2: $T(n) = \theta(n^3)$

describe at least two different conditions that could exist that would make program 2 a more efficient solution than program 1 for a specific application.

4. For each function g(n) and the time t in the following table, determine the largest size of n of a problem P that can be solved in time t if the algorithm for solving P takes g(n) microseconds (one entry is already completed).

	1 second	1 hour	1 day
n	100		
$n\log(n)$			
n^2			
n^3			
2^n			