CSCE 321, Dr. Goneid
Exercises (3)
Recursive Algorithms (Exclude & Conquer)

In the following exercises we use the General Solution of 1st Order Linear Recurrences

\[ T(n) = a_n T(n-1) + b_n , \text{ given } T(0) \text{ or } T(1) \]

with the solution:

Successive substitution gives for the cases of \( T(0) \) given and \( T(1) \) given:

\[
T(n) = T(0) \prod_{i=1}^{n} a_i + \sum_{i=1}^{n-1} b_i \prod_{j=i+1}^{n} a_j + b_n
\]

\[
T(n) = T(1) \prod_{i=2}^{n} a_i + \sum_{i=2}^{n-1} b_i \prod_{j=i+1}^{n} a_j + b_n
\]

1. Solve the following linear recurrences:
   
   a) \( T(n) = T(n-1) + 5 \) for \( n > 1 \) with \( T(1) = 0 \)
   
   b) \( T(n) = 3T(n-1) \) for \( n > 1 \) with \( T(1) = 4 \)
   
   c) \( T(n) = T(n-1) + n \) for \( n > 0 \) with \( T(0) = 0 \)
   
   d) \( T(n) = n T(n-1) \) for \( n > 0 \) with \( T(0) = 1 \)
   
   e) \( T(n) = 2n + 2 \sum_{k=1}^{n-1} T(k) \) for \( n > 1 \) with \( T(1) = 1 \)
   
   f) \( T(n) = T(n-1) + g(n) \) for \( n > 1 \) with \( T(1) = 1 \)

Consider \( g(n) \) to be: (1) \( n^2 \) (2) \( \log n \) (3) \( 2^n \)

2. Consider the following function:

```c
int task ( itemType a[ ] , int n )
{
    if ( n > 0 ) { int m = 2 * value( a , n ) + 1 ; return (m + task ( a , n-1 )) ; }
    else return 0 ;
}
```

If the function \( \text{value} (a,n) \) performs \( 4n \) arithmetic operations, what will be the exact number of such operations performed by \( \text{task} (a,n) \) as a function of \( n \)?
3. Consider the following function:

```c
float Rmul (float a[], int n)
{
    if(n<0) return 1;
    else if(a[n]==0) return 0;
    else return (a[n] * Rmul(a, n-1));
}
```

Find $T(n) =$ number of floating point multiplications as a function of $n$.

4. Consider the following function:

```c
int Foo ( itemType a[], int s, int e, int r)
{
    if ( s == e ) return 0;
    else {
        Process ( a, s, e, x, y);  // Assume this function is defined
        int d = y - x;
        return Foo ( a, s+1, e, d);  // Assume this function is defined
    }
}
```

If the above function is invoked as: $p = Foo (a, 1, n, 0)$; and the function `Process` does $2(e - s)$ arithmetic operations, construct the recurrence relation for the above algorithm, solve it and then find the exact number of arithmetic operations performed by `Foo`.

5. Consider the following function:

```c
void Doit ( itemType a[], int n, int x, int y)
{
    if ( n > 0 )
    {
        Process ( a, n, x, y);  // Assume this function is defined
        if (x > y)
            Doit ( a, n-1, x, y)
        else
            Doit ( a, n-1, y, x);
    }
}
```

If the module `Process` does $(2n-1)$ multiplications, what is the number of such operations performed by `Doit (a, n, x, y)`?
6. Consider the following function:

```c
void Doit ( itemType a[ ] , int n , int L , int H )
{
    if ( n > 0 )
    {
        Process ( a , n , L , H ) ;
        Doit ( a , n-1 , L , H ) ;
    }
}
```

If the module `Process` does \( \lceil n (n+1) / 2 \rceil \) multiplications, what is the number of such operations performed by `Doit (a , n, L , H)`?

7. Consider the following function:

```c
void Goo (int n)
{
    if ((n == 1)||(n == 2)) do 2 arithmetic operations;
    else
    {
        Goo (n-1);
        for (i = 1; i <= n; i++) do one arithmetic operation;
        Goo(n-1);
    }
}
```

Find the number \( T(n) \) of arithmetic operations done by `Goo (n)`.

8. Consider the following function:

```c
void Process (int n)
{
    if (n == 0) do one arithmetic operation;
    else
    for (i = 1; i < n; i++) Process( i );
}
```

Find the number \( T(n) \) of arithmetic operations done by `Process (n)`.

9. Consider the following recursive algorithm for computing the sum of the first \( n \) cubes: \( S(n) = 1^3 + 2^3 + \ldots + n^3 \).

```
ALGORITHM S(n)
//Input: A positive integer n
//Output: The sum of the first n cubes
if n = 1 return 1
else return S(n - 1) + n * n * n
```

Find the number \( T(n) \) of arithmetic operations done by this algorithm.
10. Consider the following recursive algorithm.

**ALGORITHM Q(n)**

//Input: A positive integer n
if n = 1 return 1
else return Q(n - 1) + 2 * n - 1

- Set up a recurrence relation for this function’s values and solve it to determine what this algorithm computes.
- Find the number T(n) of arithmetic operations done by this algorithm.

11. Find the number of moves required for a Tower of Hanoi with n discs.