EECS2040 Data Structure Hw #1 (Chapter 1, 2 of textbook)

Part 1 (40% of Hw1)

1. (20%) Using the ADT1.1 *NaturalNumber* in the textbook pp.10, add the following operations to the *NaturalNumber* ADT: Predecessor, *IsGreater*, *Multiply*, *Divide*.

Sol :

Predecessor(x) : *NaturalNumber* :: = **if**(x != 0) *Predecessor* =x - 1
else *Predecessor* = 0

IsGreater(x , y) : *Boolean* :: = **if**(x > y) *IsGreater* = Ture
else *IsGeater* = false\

Multiply(x , y) : *NaturalNumber* :: = **if**(x * y < MAXINT) *Multiply* =MAXINT
else *Multiply* = x * y

Divide(x , y) : *NaturalNumber* :: = **if**(x != 0) *Divide* = MAXINT
else *Divide* = x / y

2. (20%) Determine the frequency counts for all statements (by step table) in the following two program segments:

code (a):

```
1   for(i=1;i<=n;i++)
2       for(j=1;j<=i; j++)
3           for(k=1;k<=j;k++)
4               x++;
```

code (b)

```
1   i=1;
2   while(i<=n)
3   {
4       x++;
5       i++;
6   }
```

Sol :

coda(A)

	s/e	freq.	subtotal
for(i=1;i<=n;i++)	1	n + 1	n + 1
for(j=1;j<=i; j++)	1	$\sum_{i=1}^{n} i + 1$	$\frac{n^2 + 3n}{2}$
for(k=1;k<=j;k++)	1	$\sum_{i=1}^{n}\sum_{j=1}^{i} j + 1$	$\frac{n^3 + 6n^2 + 5n}{6}$
x++;	1	$\sum_{i=1}^{n}\sum_{j=1}^{i} j$	$\frac{n^3 + 3n^2 + 2n}{6}$

total : $\frac{n^3+6n^2+11n+3}{3}$

coda(B)

	s/e	freq.	subtotal
i=1;	1	1	1
while(i<=n)	1	n + 1	n + 1
{	0	0	0
x++	1	n	n
i++	1	n	n
}	0	0	0

total : 3n + 2

3. (20%) For the function Multiply() shown below,
   (a) Introduce statements to increment count at all appropriate points and compute the count
   (b) Simplify the resulting program by eliminating statement and compute the count
   (c) Obtain the step count for the function using the frequency method.

```
void Multiply(int **a,int **b, int **c, int m, int n, int p)
{
   for(int i=0;i<m;i++)
      for(int j=0; j<p; j++)
      {
         c[i][j] = 0;
         for(int k=0;k<n;k++)
           c[i][j] += a[i][k] * b[k][j];
      }
}
```

Sol:

(a)

```
void Multiply(int **a,int **b, int **c, int m, int n, int p)
{
   for(int i=0;i<m;i++){
         count++
      for(int j=0; j<p; j++){
         count++
         c[i][j] = 0;
         count++
         for(int k=0;k<n;k++){
              count++
              c[i][j] += a[i][k] * b[k][j];
              count++
              }
         count++
         }
        count++
      }
    count++
   }
```

(b)

```
void Multiply(int **a,int **b, int **c, int m, int n, int p)
{
   for(int i=0;i<m;i++){
      for(int j=0; j<p; j++){
         for(int k=0;k<n;k++){
           count += 2;
         }
      count += 3;
      }
   count += 2;
   }
count ++;
}
```

(c)

	s/e	freq.	subtotal
for(int i=0;i<m;i++)	1	m + 1	m + 1
for(int j=0; j<p; j++)	1	m * (p + 1)	m * (p + 1)
c[i][j] = 0;	1	m * p	m * p
for(int k=0;k<n;k++)	1	m * p * (n + 1)	m * p * (n + 1)
c[i][j] += a[i][k] * b[k][j]	1	m * p * n	m * p * n

total : 2mpn + 3mp + 2m + 1

4. (20%) A complex-valued matrix X is represented by a pair of matrices (A, B) where A and B contains real values. Write a program that computes the product of two complex-valued matrices (A, B) and (C, D), where (A, B) * (C, D) = (A+iB)*(C+iD) = (AC-BD)+i(AD + BC). Determine the number of additions and multiplications if the matrices are all nxn.

   Sol:

   一次矩陣相乘有 n^3 乘和 $n^2(n-1)$加

   一次矩陣相加有 n^2 次加

   (AC – BD) + i(AD + BC) 有 4 次矩陣相乘 2 次矩陣相加

   addition : $4n^2(n-1) + 2n^2 = 4n^3 - 2n^2$

   multiplication : $4n^3$

5. (20%) The Tower of Hanoi is a classical problem which can be solved by recurrence. There are three pegs and N disks of different sizes. Originally, all the disks are on the left peg, stacked in decreasing size from bottom to top. Our goal is to transfer all the disks to the right peg, and the rules are that we can only move one disk at a time, and no disk can be moved onto a smaller one. We can easily solve this problem with the following recursive method: If N = 1, move this disk directly to the right peg and we are done. Otherwise (N >1), first transfer the top N − 1 disks to the middle peg applying the method recursively, then move the largest disk to the right peg, and finally transfer the N −1 disks on the middle peg to the right peg applying the method recursively. Let T(N) be the total number of moves needed to transfer N disks.

   (a) Prove that T(N) = 2T(N −1) + 1 with T(1) = 1.

   (b) Unfold this recurrence relation to obtain a closed-form expression for T(N). (T(N) is expressed in terms of function of N.)

Sol :

(a)

First we move N-1 disks to the middle peg which costs T(N-1) steps,then we move the last disks from left peg to right peg which costs 1 step , finally we move the N-1 disks from the middle peg to the right peg which costs T(N-1) steps,so the sum of steps is 2T(N-1)+1.

(b)

$$T(1) = 1$$
$$T(2) = 2 * T(1) + 1$$
$$T(3) = 2 * T(2) + 1 = 2^2 * T(1) + 2 * T(1) + 1$$
$$T(4) = 2 * T(3) + 1 = 2^3 * T(1) + 2^2 * T(1) + 2^1 * T(1) + 1$$
$$\vdots$$
$$T(N) = 2^{N-1} + 2^{N-2} + 2^{N-3} + \cdots\cdots + 1 = \sum_{i=0}^{N-1} 2^i = 2^{N-1}$$

以下為和老師討論後重新計分的項目 ：

Hw1

part 1

第一題 ：

divide(x , y)在 y = 0 時 return 0 不扣分

第三題 ：

在函數最後補上 return 不扣分

```
void Multiply(int **a,int **b, int **c, int m, int n, int p)
{
   for(int i=0;i<m;i++)
      for(int j=0; j<p; j++)
      {
         c[i][j] = 0;
         for(int k=0;k<n;k++)
           c[i][j] += a[i][k] * b[k][j];
      }
   return;
}
```

第四題 ：

以自己的 code 去計算 addition 和 multiplication 數量不扣分

part 2 第三題題意不清，修改此題的評分標準
此題 35%分成三個 function 評分:

1.FastFind 11%
2.Delete 12%
3.CharDelete 12%

完成以下 function 並 demo 每個加一分

1.equality
2.empty
3.Length
4.Concat
5.substr