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

due date 3/21/2022

Format: Use a text editor to type your answers to the homework problem. You need to submit your HW in an HTML file or a DOC file named as **Hw1-SNo.doc** or **Hw1-SNo.html**, where SNo is your student number. Submit the **Hw1-SNo.doc or Hw1-SNo.html** file via eLearn. Inside the file, you need to put the **header and your student number, name (e.g., EECS2040 Data Structure Hw #1 (Chapter 1, 2 of textbook) due date 3/21/2022 by SNo, name)** first, and then the **problem** itself followed by your **answer** to that problem, one by one. The grading will be based on the correctness of your answers to the problems, and the **format**. Fail to comply with the aforementioned format (file name, header, problem, answer, problem, answer,…), will certainly degrade your score. If you have any questions, please feel free to ask.

Part 1 (2% of final Grade)

1. (10%) Using the ADT1.1 *NaturalNumber* in the textbook pp.10, add the following operations to the *NaturalNumber* ADT: Predecessor, *IsGreater*, *Multiply*, *Divide*.
2. (10%) 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   }
```

3. (10%) 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 step table method.

(d) Repeat (c) but use Θ() notation instead of exact step counts and frequency counts..

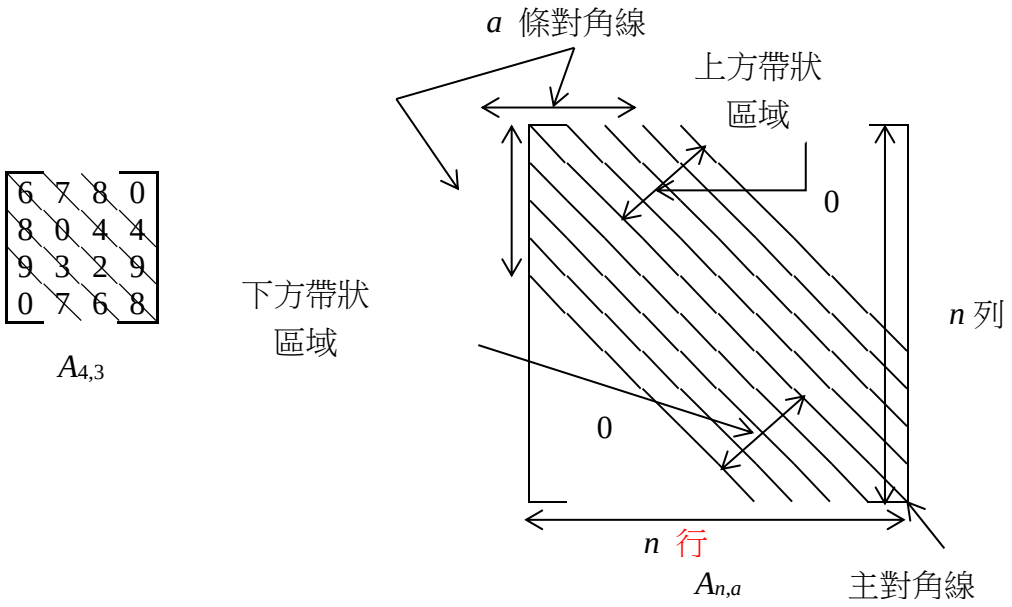
```
void Multiply(int **a, int **b, int **c, int m, int n, int p)
{
   for(int i=0;i<m;i++)
      for(int j=0; j<p; j++)
      {
         c[i][j] = 0;
         for(int k=0;k<n;k++)
           c[i][j] += a[i][k] * b[k][j];
      }
}
```

4. (10%) A complex-valued matrix X is represented by a pair of matrices (A, B) where A and B contains real values.
   (a) Write a C++ 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).
   (b) Determine the number of additions and multiplications if the matrices are all nxn.
5. (10%) 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.)

6. (10%) For the polynomial represented by dynamic array of tuples,
   (a) Design an algorithm for performing multiplication of two polynomials.

(b) Then analyze its time complexity using Θ() notation, assuming the number of terms of the two polynomials are m and n, respectively.

7. (10%) Obtain an addressing formula for the element a[i1][i2]…[in] in an array declared as a[u1][u2]…[un]. Assume a column-major representation of the array with one word per element and a the address of a[0][0]…[0]. In a column-major representation, a two-dimensional array is atored sequentially by columns rather than by rows.
8. (15%) A square band matrix $A_{n,a}$ is an n x n matrix A in which all the nonzero terms lie in a band centered around the main diagonal. The band includes a-1 diagonals below and above the main diagonal as shown below.



(a) How many elements are there in the band of $A_{n,a}$?

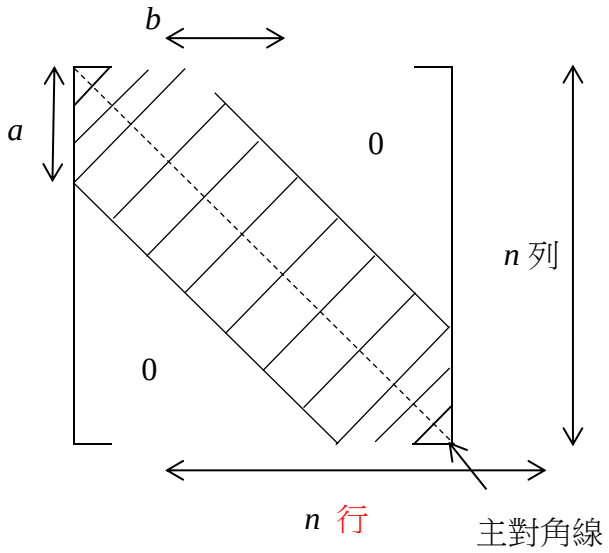
(b) What is the relationship between i and j for element A_{ij} in the band of $A_{n,a}$?

(c) Assume that the band of An,a is stored sequentially in an array b by diagonals starting with the lowermost diagonal. Thus $A_{4,3}$ above would have the following representation:

b[0]	b[1]	b[2]	b[3]	b[4]	b[5]	b[6]	b[7]	b[8]	b[9]	b[10]	b[11]	b[12]	b[13]
9	7	8	3	6	6	0	2	8	7	4	9	8	4
A_{20}	A_{31}	A_{10}	A_{21}	A_{32}	A_{00}	A_{11}	A_{22}	A_{33}	A_{01}	A_{12}	A_{23}	A_{02}	A_{13}

Obtain an addressing formula for the location of an element A_{ij} in the lower band of $A_{n,a}$, e.g., LOC(A_{20}) = 0, LOC(A_{31}) = 1 in the example above.

9. (15%) A generalized band matrix $A_{n,a,b}$ is an n x n matrix A in which all the nonzero terms lie in a band made up of a-1 diagonals below the main diagonal, the main diagonal, and b-1 above the main diagonal as shown below.



$A_{n,\,a,\,b}$

(a) How many elements are there in the band of $A_{n,a,b}$?

(b) What is the relationship between i and j for element A_{ij} in the band of $A_{n,a,b}$?

(c) Obtain a sequential representation of the band of $A_{n,a,b}$ in one-dimensional array c.