EECS2040 Data Structure Hw #1 (Chapter 1, 2 of textbook)
due date 3/27/2022

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 4/8/2022 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 2 Coding (5% of final Grade)

You should submit:
(a) All your source codes (C++ file).
(b) Show the execution trace of your program.

1. (30%) Write a C++ program to implement the **ADT2.3 Polynomial** below using Representation 3 (dynamic array of (coef, exp) tuples).

```
class Polynomial {
// p(x) = a0 x^e0 + … + an x^en
// where ai is nonzero float and ei is non-negative int
public:
    Polynomial( );
    //construct the polynomial p(x) = 0
    Polynomial Add(Polynomial poly);
    //return the sum of *this and poly
    Polynomial Mult(Polynomial poly);
    //return the product of *this and poly
    float Eval(float f );
    //Evaluate the polynomial *this at f and return the results
    int operator!();
        // if *this is the zero polynomial, return 1; else return 0;
};
```

Implement the Mult(Polynomial p) and Eval(float x). Add four more functions: two to input and output polynomials via **overloading** the >> and << **operators**. And

```
Coefficient Coef(Exponent e);
                    // return the coefficient of e in *this
Exponent LeadExp();
                  // return the largest exponent in *this
```

Where Coefficient denotes the type of coefficient, usually float, and Exponent denotes the type of exponent, usually int.

You should try out at least two runs of your program (execution trace) to **demonstrate** the Add, Mult, Eval and input, output functions.

2. (35%) Write a C++ program to implement the **ADT2.4 SparseMatrix** in textbook shown below (with Transpose implemented by FastTranspose).

```
class SparseMatrix
{//三元組，<列，行，值>，的集合，其中列與行為非負整數，
 //並且它的組合是唯一的；值也是個整數。
public:
    SparseMatrix(int r, int c, int t);
    //constructor.
    //r is #row, c is #col, t is #non-zero terms
    SparseMatrix Transpose( );
    //回傳將 *this 中每個三元組的行與列交換後的 SparseMatrix
    SparseMatrix Add(SparseMatrix b);
    // 如果 *this 和 b 的維度一樣，那麼就把相對應的項給相加，
    // 亦即，具有相同列和行的值會被回傳；否則的話丟出例外。
    SparseMatrix Multiply(SparseMatrix b);
// 如果*this 中的行數和 b 中的列數一樣多的話，那麼回傳的矩陣 d= *this 和 b
//（依據 d[i][j]=Σ(a[i][k]・b[k][j]，其中 d[i][j]是第 (i,j) 個元素）相乘的結果。
// k 的範圍從 0 到*this 的行數減 1；如果不一樣多的話，那麼就丟出例外。
};
```

You should build you program based on the example codes in the book and implement the **Add** function and functions to **input**, **output** a sparse matrix by **overloading** the >> and << **operators**.

You should try out at least two runs of your program to demonstrate the Add, Mult, FastTranspose, and input, output functions.

3. (35%) Write a C++ program to implement the **ADT2.5 String** (with Find function implemented by FastFind).

```
class String
{
public:
    String(char *init, int m);
    // constructor using input string init of length m
    bool operator == (String t);//equality test
    bool operator!( ); // empty test, true or false
    int Length( );//get the number of characters of *this
    String Concat(String t);
    // concatenation with another string t
    String Substr(int i, int j);    // generate a substring i~j-1
    int Find(String pat);
    // Return an index i such that pat matches the substring
    // of the object begins at position i.    Return -1 if pat
    // is empty or not a substring of the object
}
```

In addition, write **three more functions**:

String String::Delete(int start, int length); //remove length characters beginning at start

String String::CharDelete(char c); //returns the string with all occurrence of c removed.

int String::Compare(String y); //compare two strings of letters of alphabet.

If two strings of letter of alphabet, $x = (x_0,\ldots,x_{m-1})$ and $y=(y_0,\ldots,y_{n-1})$ where x_i, y_j are letters, then the Compare member function will decide whether $x<y$, $x=y$, or $x>y$, where $x < y$ means if $x_i=y_i$ for $0\le i<j$ and $x_j<y_j$ or if $x_i=y_i$ for $0\le i\le m$ and $m<n$.

$x=y$ means $m=n$ and $x_i=y_i$ for $0\le i<n$.

$x>y$ means if $x_i=y_i$ for $0\le i<j$ and $x_j>y_j$ or if $x_i=y_i$ for $0\le i\le n$ and $m>n$.

The Compare function will return either -1, 0, or +1 if $x<y$, $x=y$, or $x>y$, respectively.

You should try out at least two runs of your program to demonstrate all those functions.