**EECS2040 Data Structure Hw #1\_Part2 (Chapter 1, 2 of textbook)**

**due date 3/27/2022**

**BY 108011235 陳昭維**

**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.

**<Question>**

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**

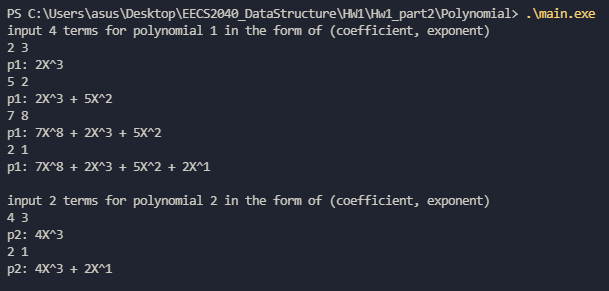
Exponent LeadExp();

// 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.

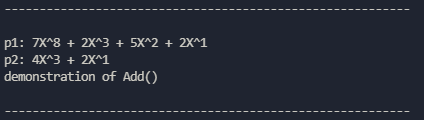
**<Answers – Trial 1>**

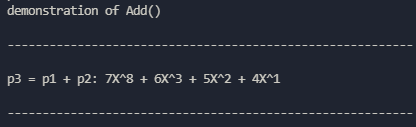
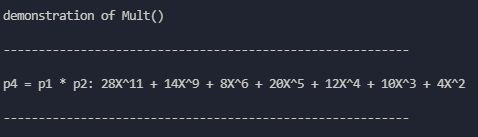
****

Uses overloaded operator>> and operator<< to input polynomial terms and output polynomial.

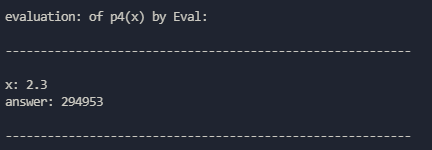
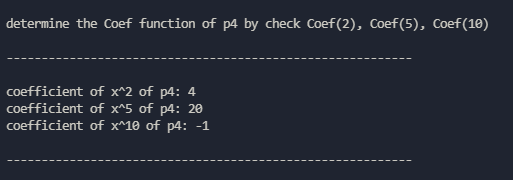
I default set 4 terms for p1 and 2 terms for p2, and randomly input (coef, exp) number.

My code has ability to sort, thus regardless of the input order, the highest exponent will be at front.

****

** **

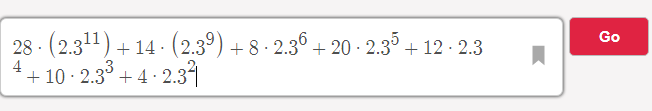
From above, we can calculate the addition of and multiplication product

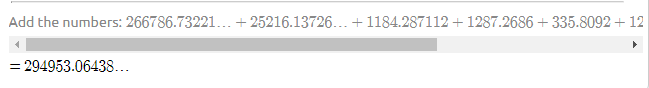
** **

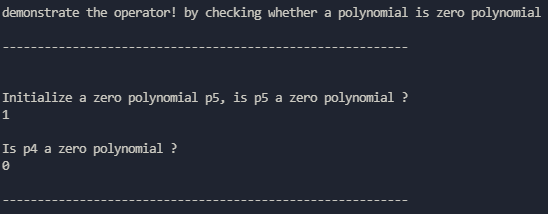
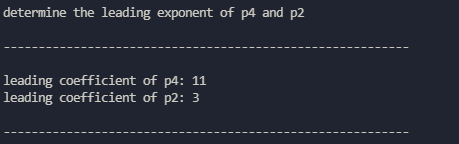
Plug in x = 2.3 to the polynomial p4,

Uses the Coef() function to return the coefficient for given exponent, if exponent not found, then return -1.

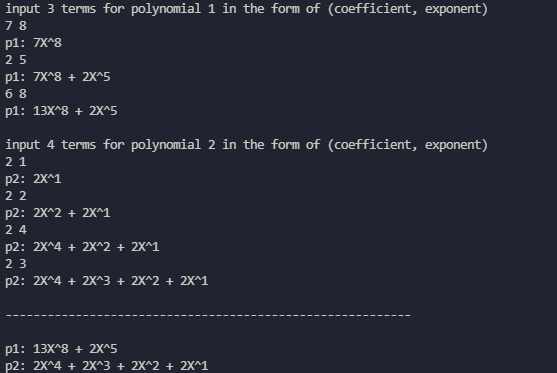
Uses the Lead() function to find the leading coefficient





****

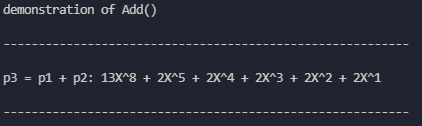
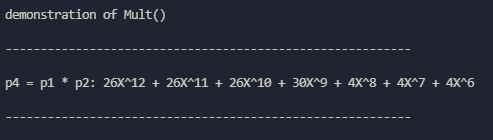
**<Answers – Trial 2>**

****

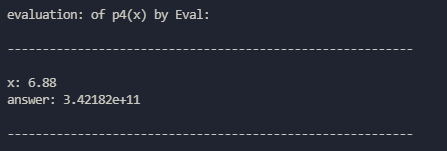
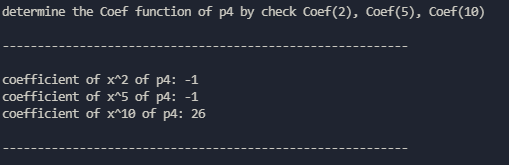
Uses overloaded operator>> and operator<< to input polynomial terms and output polynomial.

I default set 3 terms for p1 and 4 terms for p2, and randomly input (coef, exp) number.

Also, my code has ability to add onto existed exponent, so for p1 although 3 terms input, but there is 2 term due to same exponent.

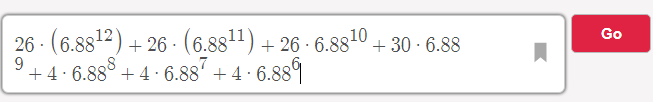
**** ****

From above, we can calculate the addition of and multiplication product

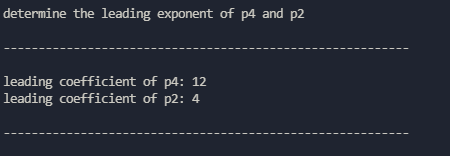
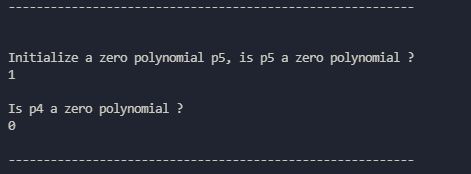
**** ****

Uses the Coef() function to return the coefficient for given exponent, if exponent not found, then return -1.

Uses the Lead() function to find the leading coefficient





**** ****

**<Question>**

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。

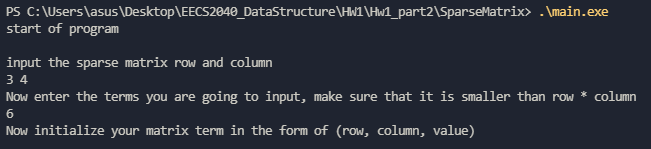
// *k*的範圍從0到\***this**的行數減1；如果不一樣多的話，那麼就丟出例外。

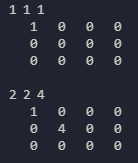
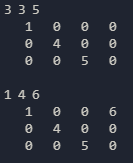
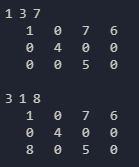
};

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.

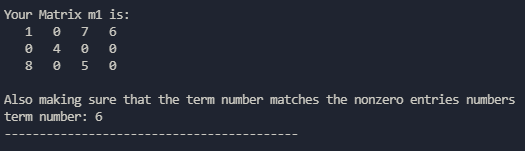
**<Answer – Trial 1>**

****

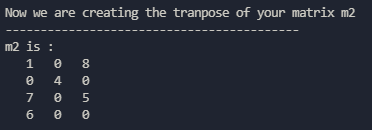
**  **

Uses the operator >> to input matrix term in the form of (row, col, value)

And then immediately output the sparse matrix by using operator<<

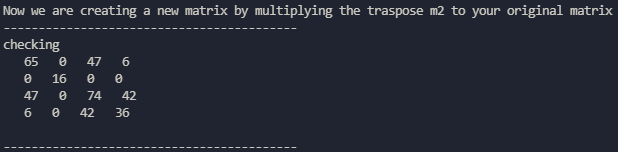
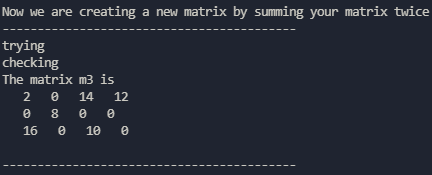
** **

Uses the self-defined getTerm function to make sure that the term number matches the nonzero entries.

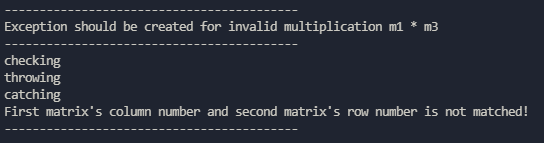
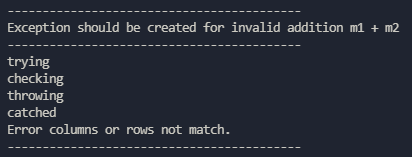
** **

Uses the transpose function to create the transpose matrix of m1

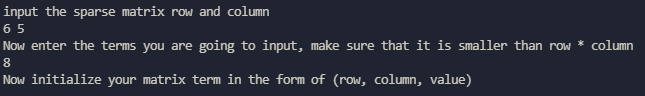
**Test for the Add function and Multiply function**

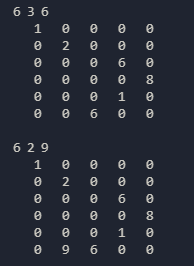
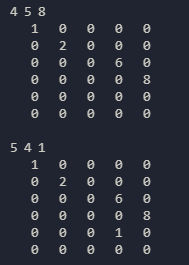
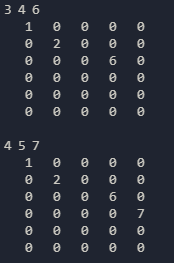
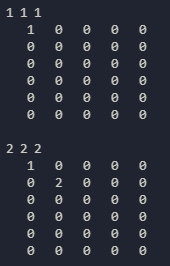
****

**Test for exception when invalid operation happens**

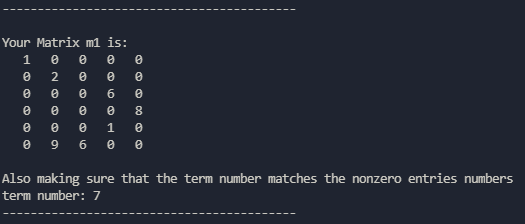
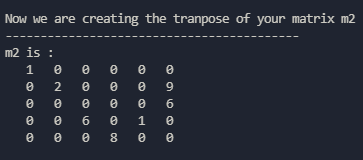
****

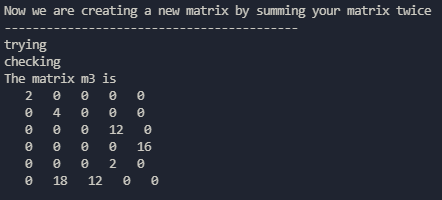
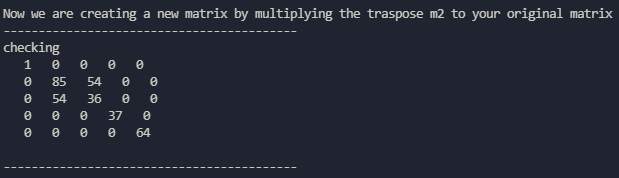
**<Answer – Trial 2>**

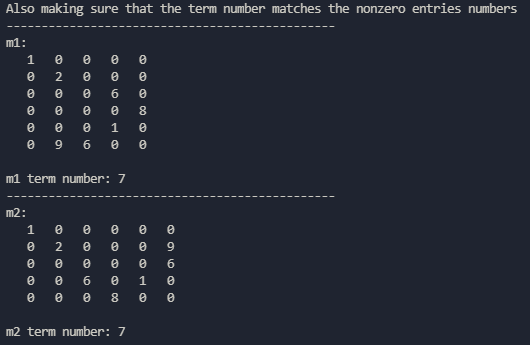
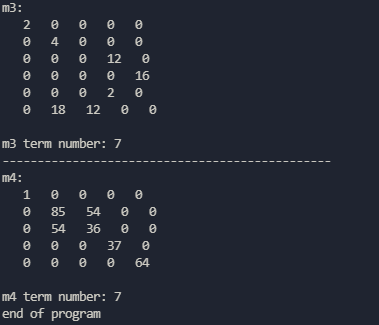
****

****

replaced

** **

** **

** **

**<Question>**

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.

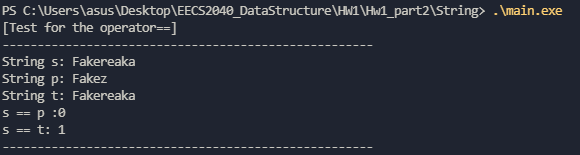
If two strings of letter of alphabet, x = (x0,…,xm-1) and y=(y0,…,yn-1) where xi, yj are letters, then the Compare member function will decide whether x<y, x=y, or x>y, where x < y means if xi=yi for 0≤i<j and xj<yj or if xi=yi for 0≤i≤m and m<n. x=y means m=n and xi=yi for 0≤i<n.

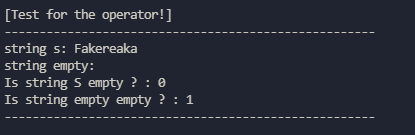
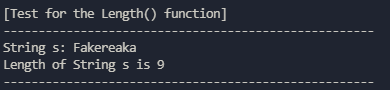
x>y means if xi=yi for 0≤i<j and xj>yj or if xi=yi for 0≤i≤n and m>n.

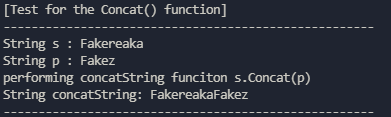
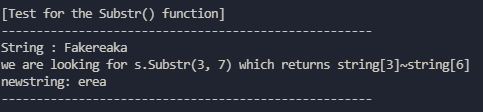
The Compare function will return either -1, 0, or +1 if x<y, x=y, or x>y, respectively.

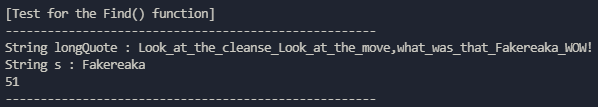
You should try out at least two runs of your program to demonstrate all those functions.

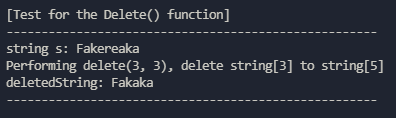
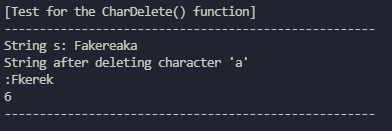
**<Answer-Trial 1>**

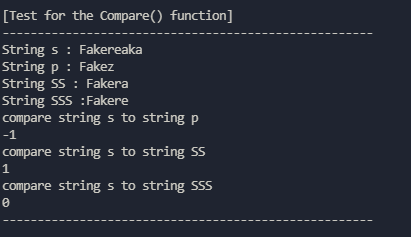
****

** **

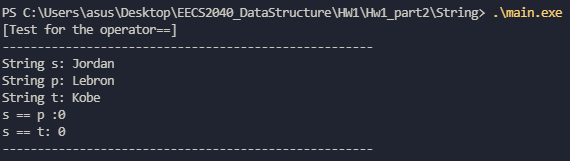
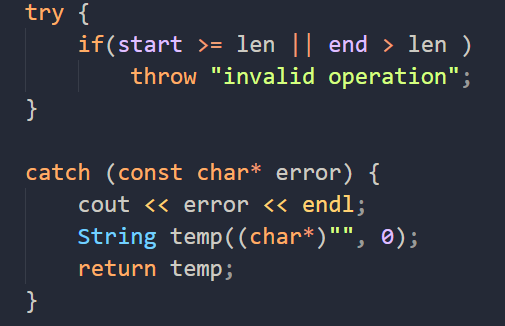
** **

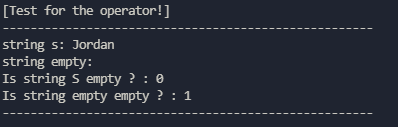
****

** **

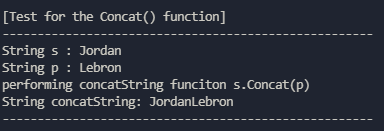
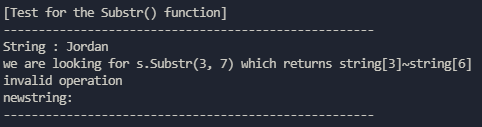
****

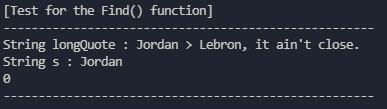
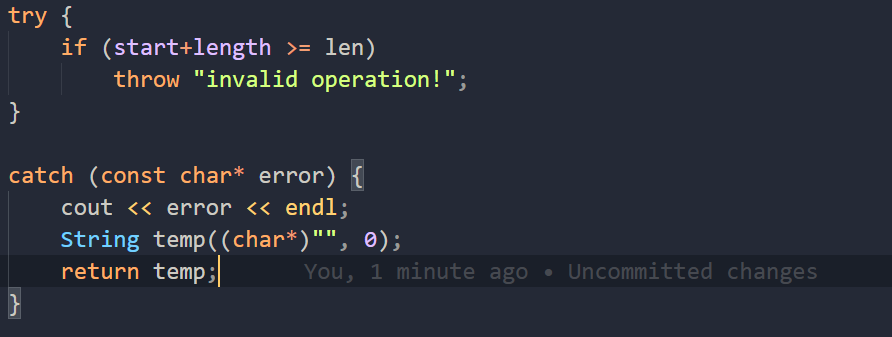
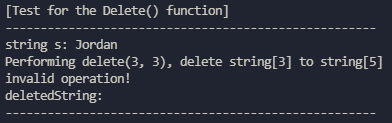
**<Answer-Trial 2>**

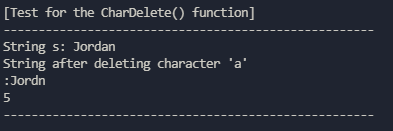
****

****

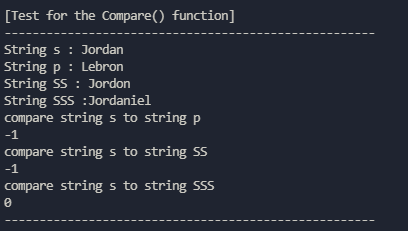
Return empty string for invalid operation

** **

** **

****

Return empty string for invalid operation

****