EECS2040 Data Structure Hw #3 (Chapter 4 Linked List)
due date 4/25/2021

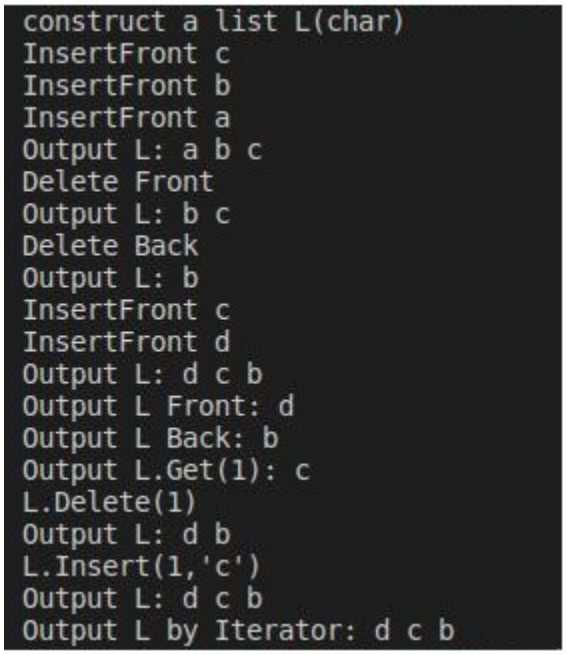
Format: Use a text editor to type your answers to the homework problem. You need to submit your HW in an HTML file or a DOC file named as **Hw3-SNo.doc** or **Hw3-SNo.html**, where SNo is your student number. Send the **Hw3-SNo.doc or Hw3-SNo.html** file to me (tljong@mx.nthu.edu.tw) via e-mail. Inside the file, you need to put the **header and your student number, name (e.g., EE2410 Data Structure Hw #3 (Chapter 4 of textbook) due date 4/25/2021 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 me.

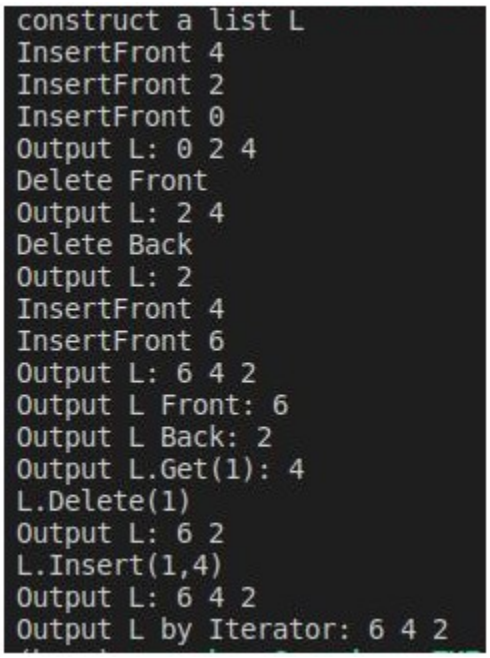
Part 2 Coding (due 5/8)

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

1. (30%) Fully code and test the C++ template class List<T> shown in Part 1 Problem 3 above. You must include:
   a. A constructor which constructs an initially empty list.
   b. A destructor which deletes all nodes in the list.
   c. InsertFront() function to insert at the front of the list.
   d. DeleteFront() and DeleteBack() to delete from either end.
   e. Front() and Back() functions to return the first and last elements of the list, respectively.
   f. A function Get(int i) that returns the ith element in the list.
   g. Delete(int i) to delete the ith element
   h. Insert(int i, T e) to insert as the ith element
   i. Overload the output operator << to out put the List object.
   j. As well as functions and forward iterator as shown above.

   Write a client program (main()) to **demonstrate** those functions you developed.
   利用 Insert(),InsertFront()來進行插入，用 Delete(),DeleteFront()刪除元素，用 Get(),Front(),Back()得到不同位置元素的值，用<< ,>>輸出和輸入。





2. (35%) Develop a C++ class Polynomial to represent and manipulate univariate polynomials with double coefficients (use circular linked list with header nodes). Each term of the polynomial will be represented as a node. Thus a node in this system will have three data members as below.

coef	exp	link

Each polynomial is to be represented as a circular list with header node. To delete polynomials efficiently, we need to use an **available-space list** and associated functions GetNode() and RetNode() described in Section 4.5. The external (i.e., for input and output) representation of a univariate polynomial will be assumed to be a sequence of integers and doubles of the form: n, c_1, e_1, c_2, e_2, c_3, e_3,…, c_n, e_n, where e_i represents an integer exponent and c_i a double coefficient; n gives the number of terms in the polynomial. The exponents of the polynomial are in decreasing order.

Write and **test** the following functions:

(a) Istream& operator>>(istream& is, Polynomial& x): Read in an input polynomial and convert it to its circular list representation using a header node.
(b) Ostream& operator<<(ostream& os, Polynomial& x): Convert x from its linked list representation to its external representation and output it.
(c) Polynomila::Polynomial(const Polynomial& a): copy constructor
(d) Const Polynomila& Polynomial::operator=(const Polynomial& a) const[assignment operator]: assign polynomial a to *this.
(e) Polynomial::~ Polynomial(): desctructor, return all nodes to available-space list
(f) Polynomial operator+ (const Polynomial& b) const: Create and return the polynomial *this + b
(g) Polynomial operator- (const Polynomial& b) const: Create and return the polynomial *this – b
(h) Polynomial operator* (const Polynomial& b) const: Create and return the polynomial *this * b
(i) double Polynomial::Evaluate(double x) const: Evaluate the polynomial *this and return the result.

用<<,>>來輸出輸入，用 Polynomial()作為 copy constructor， 用~Polynomial()作為 destructor()，用+-* 進行多項式間的運算，用 evalute()得到多項式的變數帶入特定值的結果，輸入時由高次項到低次項輸入。

```
input a(高次到低次)
2
2 2
1 1
output a
2x^2+1x^1
copy a to c by Polynomial(const Polynomial& a)
output c
2x^2+1x^1
assign a to d by =
output d
2x^2+1x^1
input b
2
1 2
2 1
output b
1x^2+2x^1
output a+b
3x^2+3x^1
output a-b
1x^2+-1x^1
output a*b
2x^4+5x^3+2x^2
output a.evaluate(2)
10
```

```
input a(高次到低次)
3
1 2
-1 1
1 0
output a
1x^2+-1x^1+1
copy a to c by Polynomial(const Polynomial& a)
output c
1x^2+-1x^1+1
assign a to d by =
output d
1x^2+-1x^1+1
input b
2
1 1
1 0
output b
1x^1+1
output a+b
1x^2+2
output a-b
1x^2+-2x^1
output a*b
1x^3+1
output a.evaluate(2.5)
4.75
```

3. (35%) The class definition for sparse matrix in Program 4.29 is shown below.

```
struct Triple{int row, col, value;};
class Matrix; // 前向宣告
class MatrixNode {
friend class Matrix;
friend istream& operator>>(istream&, Matrix&); // 為了能夠讀進矩陣
private:
   MatrixNode *down , *right;
   bool head;
   union { // 沒有名字的 union
      MatrixNode *next;
      Triple triple;
   };
   MatrixNode(bool, Triple*); // 建構子
}

MatrixNode::MatrixNode(bool b, Triple *t)  // 建構子
```

```
{
   head = b;
   if (b) {right = down = this;} // 列/行的標頭節點
   else triple = *t; // 標頭節點串列的元素節點或標頭節點
}

class Matrix{
friend istream& operator>>(istream&, Matrix&);
public:
   ~Matrix(); // 解構子
private:
   MatrixNode *headnode;
};
```

Based on this class, do the following tasks.

(a) Write the C++ function, **operator**+(**const** Matrix& b) **const**, which returns the matrix ***this** + b.

(b) Write the C++ function, **operator***(const Matrix& b) **const**, which returns the matrix ***this** * b.

(c) Write the C++ function, **operator**<<(), which outputs a sparse matrix as triples (i, j, a_{ij}).

(d) Write the C++ function, Transpose(), which transpose a sparse matrix.

(e) Write and test a copy constructor for sparse matrices. What is the computing time of your copy constructor?

Write a client program (main()) to **demonstrate** those functions you developed.

用+* 來進行兩個矩陣間的運算，用>>,<<輸入和輸出，用 Transpose()得到轉置矩陣，用 Matrix()為 copy constructor 將一個矩陣的值 copy 到另一個。
我所寫的 copy constructor 跟據執行的步驟大致要將整個矩陣的 node 都經過一遍，所以 time complexity 為 $\Theta(row.col)$。

```
input matrix a
input dimension
2 2 4
input node
0 0 1
0 1 2
1 0 3
1 1 4
output matrix a
(0 0 1)
(0 1 2)
(1 0 3)
(1 1 4)
input matrix b
input dimension
2 2 4
input node
0 0 4
0 1 3
1 0 2
1 1 1
output matrix b
(0 0 4)
(0 1 3)
(1 0 2)
(1 1 1)
output matrix a+b
(0 0 5)
(0 1 5)
(1 0 5)
(1 1 5)
output matrix a*b
(0 0 8)
(0 1 5)
(1 0 20)
(1 1 13)
output matrix a*Transpose()
(0 0 1)
(0 1 3)
(1 0 2)
(1 1 4)
copy a to c by copy constructor
output c
(0 0 1)
(0 1 2)
(1 0 3)
(1 1 4)
```

```
input matrix a
input dimension
3 3 4
input node
0 0 1
0 1 2
1 1 3
2 2 8
output matrix a
(0 0 1)
(0 1 2)
(1 1 3)
(2 2 8)
input matrix b
input dimension
3 3 4
input node
0 0 2
0 1 3
1 1 44
2 2 18
output matrix b
(0 0 2)
(0 1 3)
(1 1 44)
(2 2 18)
output matrix a+b
(0 0 3)
(0 1 5)
(1 1 47)
(2 2 26)
output matrix a*b
(0 0 2)
(0 1 91)
(1 1 132)
(2 2 144)
output matrix a*Transpose()
(0 0 1)
(1 0 2)
(1 1 3)
(2 2 8)
copy a to c by copy constructor
output c
(0 0 1)
(0 1 2)
(1 1 3)
(2 2 8)
```