EE2410 Data Structure Coding HW #3 (Chapter 4 of textbook)

due date 5/5/2024(Sun.) 23:59

You should submit:

(a) All your source codes (C++ file).

(b) Show the execution trace of your program, i.e., write a client main() to demonstrate all functions you designed using example data.

Submit your homework before the deadline (midnight of 5/5). Fail to comply (**late** homework) will have ZERO score. **Copy** homework will have ZERO score on both parties and SERIOUS consequences.

1. (25%) template Chain

Given a template linked list **L** instantiated by the Chain class with a pointer **first** to the first node and **last** to the last node of the list as shown below. The node is a ChainNode object consisting of a template data and link field.

```
template < class T > class Chain;    // 前向宣告
template < class T >class ChainNode {
friend class Chain <T>;
private:
  T data;
  ChainNode<T>* link;
};

template <class T>class Chain {
public:
  Chain( ) {first = last = 0;} //  建構子將 first, last 初始化成 0
  ~Chain(); //desctructor
// 鏈的處理運算
  bool IsEmpty();
  int Size();
  void InsertHead(const T& e);
  void DeleteHead();
  const T& Front();
  const T& Back();
  void InsertBack(const T& e);
  void DeleteBack();
```

```
 T& Get(int index);
 T& Set(int index, const T& e);
 int IndexOf(const T& e) const;
 void Delete(int index);
 void Insert(int index, const T& e);
 void Concatenate(Chain<T>& b);
 void Reverse();
 void Delete(Position p);
 void Insert(Position p, const T& e); //Position means ChainNode*)
 class ChainIterator{
    public:
       .....
   };
 ChainIterator begin() {return ChainIterator(first);}
 ChainIterator end()   {return ChainIterator(0);}
private:
 ChainNode<T> * first, *last;
};
```

(a) **Implement the above Chain ADT.**

(b) Overload the output operator << to output all elements of the List object.

(c) **A member function** that will perform an insertion to the **immediate before of the kth node** in the list L.

(d) **A member function** that will **delete every other node** of L beginning with node first (i.e., the first, 3rd, 5th,…nodes of L are deleted).

(e) **A member function** divideMid that will divides the given list into two sublists of (almost) equal sizes. Suppose myList points to the list with elements 34 65 27 89 12 (in this order). The statement: myList.divideMid(subList); divides myList into two sublists: myList points to the list with the elements 34 65 27, and subList points to the sublist with the elements 89 12. Formulate a step-by-step algorithm to perform this task.

(f) **A member function deconcatenate(**ChainNode* p) that will (or **split**) a linked list L into two linked list. Assume the node denoted by the pointer variable split is to be the first node in the returned second linked list.

(g) Assume L_1 and L_2 are two chains: $L_1 = (x_1,x_2,..,x_n)$ and $L_2 = (y_1,y_2,\dots,y_m)$, respectively. **Add a member function** that can **merge** the two chains together to obtain the chain $L_3 = (x_1,y_1,x_2,y_2,\dots,x_m,y_m,x_{m+1},..,x_n)$ if n>m and $L_3 = (x_1,y_1,x_2,y_2,\dots,x_n,y_n,y_{n+1},..,y_m)$ if n<m.

(h) Implement the stack data structure as a derived class of the class Chain<T>.

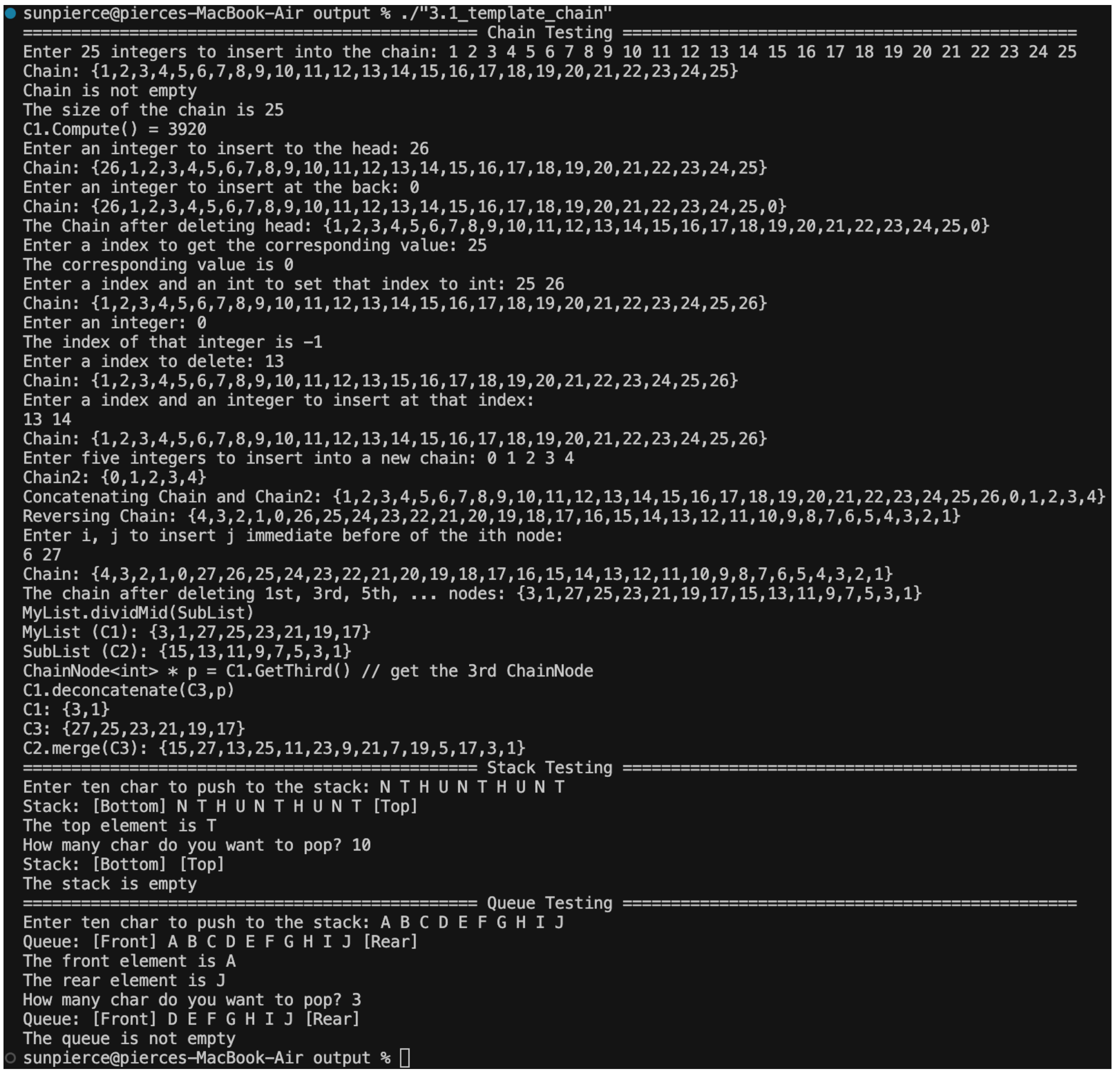
(i) Implement the queue data structure as a derived class of the class Chain<T>.

(j) Let $x_1, x_2, \ldots, x_n$ be the elements of a Chain<int> object. Each x_i is an integer. Write C++ code to compute the expression $\sum_{i=1}^{n-5}(x_i \times x_{i+5})$

Write a client program (main()) to **demonstrate** those functions you developed. Use an int list consists of {1,2,3,..,25} 25 integers as example.

sol:

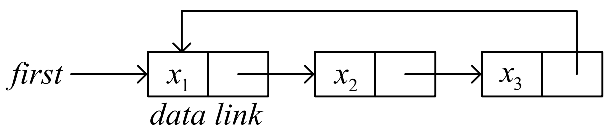
Execution trace:



```
sunpierce@pierces-MacBook-Air output % ./"3.1_template_chain"
=============================================== Chain Testing ===============================================
Enter 25 integers to insert into the chain: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
Chain: {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25}
Chain is not empty
The size of the chain is 25
C1.Compute() = 3920
Enter an integer to insert to the head: 26
Chain: {26,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25}
Enter an integer to insert at the back: 0
Chain: {26,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0}
The Chain after deleting head: {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0}
Enter a index to get the corresponding value: 25
The corresponding value is 0
Enter a index and an int to set that index to int: 25 26
Chain: {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26}
Enter an integer: 0
The index of that integer is -1
Enter a index to delete: 13
Chain: {1,2,3,4,5,6,7,8,9,10,11,12,13,15,16,17,18,19,20,21,22,23,24,25,26}
Enter a index and an integer to insert at that index:
13 14
Chain: {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26}
Enter five integers to insert into a new chain: 0 1 2 3 4
Chain2: {0,1,2,3,4}
Concatenating Chain and Chain2: {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,0,1,2,3,4}
Reversing Chain: {4,3,2,1,0,26,25,24,23,22,21,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1}
Enter i, j to insert j immediate before of the ith node:
6 27
Chain: {4,3,2,1,0,27,26,25,24,23,22,21,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1}
The chain after deleting 1st, 3rd, 5th, ... nodes: {3,1,27,25,23,21,19,17,15,13,11,9,7,5,3,1}
MyList.dividMid(SubList)
MyList (C1): {3,1,27,25,23,21,19,17}
SubList (C2): {15,13,11,9,7,5,3,1}
ChainNode<int> * p = C1.GetThird() // get the 3rd ChainNode
C1.deconcatenate(C3,p)
C1: {3,1}
C3: {27,25,23,21,19,17}
C2.merge(C3): {15,27,13,25,11,23,9,21,7,19,5,17,3,1}
=============================================== Stack Testing ===============================================
Enter ten char to push to the stack: N T H U N T H U N T
Stack: [Bottom] N T H U N T H U N T [Top]
The top element is T
How many char do you want to pop? 10
Stack: [Bottom] [Top]
The stack is empty
=============================================== Queue Testing ===============================================
Enter ten char to push to the stack: A B C D E F G H I J
Queue: [Front] A B C D E F G H I J [Rear]
The front element is A
The rear element is J
How many char do you want to pop? 3
Queue: [Front] D E F G H I J [Rear]
The queue is not empty
sunpierce@pierces-MacBook-Air output % 
```

2. (25%) circular linked list

Given a **circular linked list L** instantiated by **class** CircularList containing a private data member, **first** pointing to the first node in the circular list as shown below



A circular linked list

Write C++ codes to

(a) count the number of nodes in the circular list.

(b) insert a new node at the front of the list InsertFront().

(c) insert a new node at the back (right after the last node) of the list InsertBack().

(d) delete the first node of the list DeleteFirst().

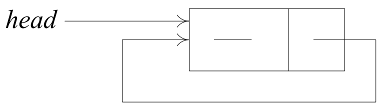
(e) delete the last node of the list DeleteBack().

(f) **delete every other node** of the list beginning with node first (i.e., the first, 3rd, 5th,…nodes of L are deleted).

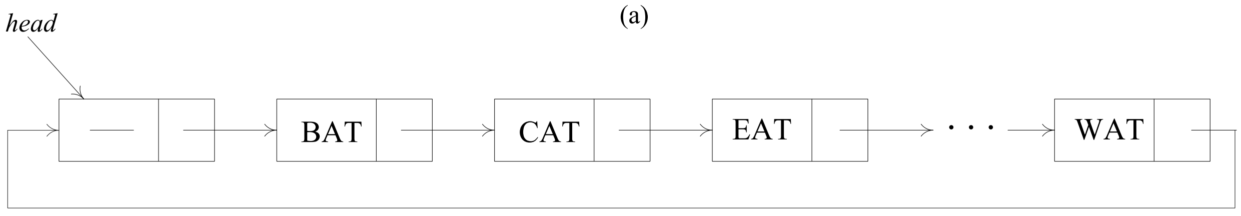
(g) **deconcatenate** (or **split**) a linked circular list L into two circular lists. Assume the node denoted by the pointer variable split is to be the first node in the second circular list.

(h) Assume L_1 and L_2 are two circular lists: $L_1 = (x_1,x_2,..,x_n)$ and $L_2 = (y_1,y_2,\dots,y_m)$, respectively. Implement a member function that can **merge** the two chains together to obtain the chain $L_3 = (x_1,y_1,x_2,y_2,\dots,x_m,y_m,x_{m+1},..,x_n)$ if n>m and $L_3 = (x_1,y_1,x_2,y_2,\dots,x_n,y_n,y_{n+1},..,y_m)$ if n<m.

(i) (15%) Repeat (a) – (h) above if the circular list is modified as shown in Figure 4.16 below by introducing a dummy node, header.



(a)

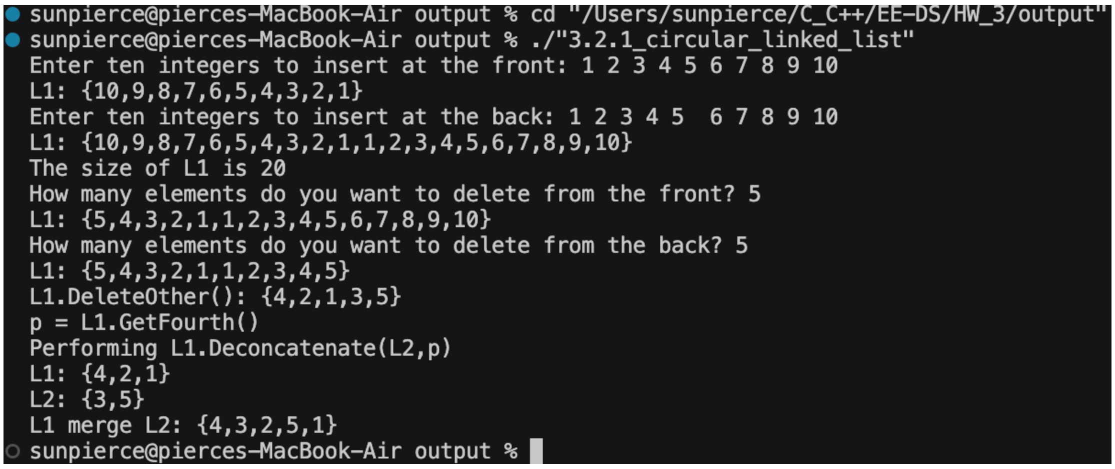


(b)

Figure 4.16 Circular list with a header node

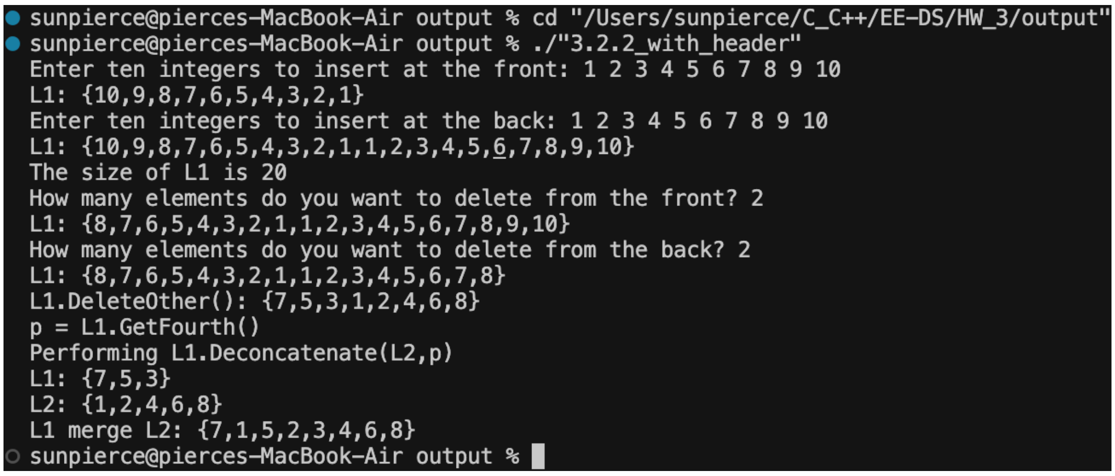
Write a client program (main()) to **demonstrate** those functions you developed.

```
sol:
Execution trace:
w/o header:
```



```
sunpierce@pierces-MacBook-Air output % cd "/Users/sunpierce/C_C++/EE-DS/HW_3/output"
sunpierce@pierces-MacBook-Air output % ./"3.2.1_circular_linked_list"
 Enter ten integers to insert at the front: 1 2 3 4 5 6 7 8 9 10
 L1: {10,9,8,7,6,5,4,3,2,1}
 Enter ten integers to insert at the back: 1 2 3 4 5  6 7 8 9 10
 L1: {10,9,8,7,6,5,4,3,2,1,1,2,3,4,5,6,7,8,9,10}
 The size of L1 is 20
 How many elements do you want to delete from the front? 5
 L1: {5,4,3,2,1,1,2,3,4,5,6,7,8,9,10}
 How many elements do you want to delete from the back? 5
 L1: {5,4,3,2,1,1,2,3,4,5}
 L1.DeleteOther(): {4,2,1,3,5}
 p = L1.GetFourth()
 Performing L1.Deconcatenate(L2,p)
 L1: {4,2,1}
 L2: {3,5}
 L1 merge L2: {4,3,2,5,1}
sunpierce@pierces-MacBook-Air output %
```

w/ header:



```
sunpierce@pierces-MacBook-Air output % cd "/Users/sunpierce/C_C++/EE-DS/HW_3/output"
sunpierce@pierces-MacBook-Air output % ./"3.2.2_with_header"
 Enter ten integers to insert at the front: 1 2 3 4 5 6 7 8 9 10
 L1: {10,9,8,7,6,5,4,3,2,1}
 Enter ten integers to insert at the back: 1 2 3 4 5 6 7 8 9 10
 L1: {10,9,8,7,6,5,4,3,2,1,1,2,3,4,5,6,7,8,9,10}
 The size of L1 is 20
 How many elements do you want to delete from the front? 2
 L1: {8,7,6,5,4,3,2,1,1,2,3,4,5,6,7,8,9,10}
 How many elements do you want to delete from the back? 2
 L1: {8,7,6,5,4,3,2,1,1,2,3,4,5,6,7,8}
 L1.DeleteOther(): {7,5,3,1,2,4,6,8}
 p = L1.GetFourth()
 Performing L1.Deconcatenate(L2,p)
 L1: {7,5,3}
 L2: {1,2,4,6,8}
 L1 merge L2: {7,1,5,2,3,4,6,8}
sunpierce@pierces-MacBook-Air output %
```

3. (20%) Linked polynomial

Develop a C++ class Polynomial to represent and manipulate univariate polynomials with double-type 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

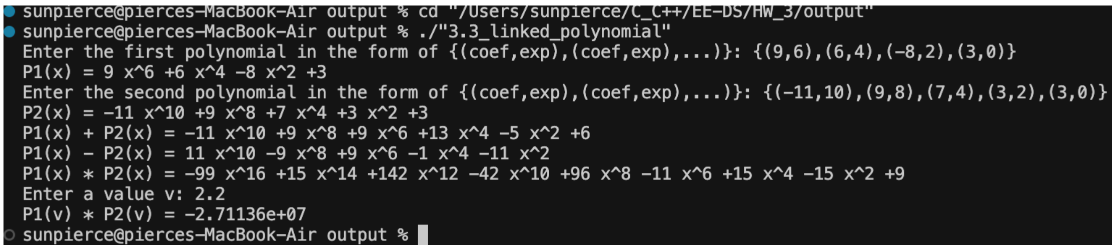
(h) Polynomial operator* (const Polynomial& b) const: Create and return the polynomial *this * b

(i) double Polynomial::Evaluate(double x) const: Evaluate the polynomial *this and return the result.

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

sol:

Execution trace:



```
sunpierce@pierces-MacBook-Air output % cd "/Users/sunpierce/C_C++/EE-DS/HW_3/output"
sunpierce@pierces-MacBook-Air output % ./"3.3_linked_polynomial"
Enter the first polynomial in the form of {(coef,exp),(coef,exp),...)}: {(9,6),(6,4),(-8,2),(3,0)}
P1(x) = 9 x^6 +6 x^4 -8 x^2 +3
Enter the second polynomial in the form of {(coef,exp),(coef,exp),...)}: {(-11,10),(9,8),(7,4),(3,2),(3,0)}
P2(x) = -11 x^10 +9 x^8 +7 x^4 +3 x^2 +3
P1(x) + P2(x) = -11 x^10 +9 x^8 +9 x^6 +13 x^4 -5 x^2 +6
P1(x) - P2(x) = 11 x^10 -9 x^8 +9 x^6 -1 x^4 -11 x^2
P1(x) * P2(x) = -99 x^16 +15 x^14 +142 x^12 -42 x^10 +96 x^8 -11 x^6 +15 x^4 -15 x^2 +9
Enter a value v: 2.2
P1(v) * P2(v) = -2.71136e+07
sunpierce@pierces-MacBook-Air output %
```

4. (20%) linked sparse matrix

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.

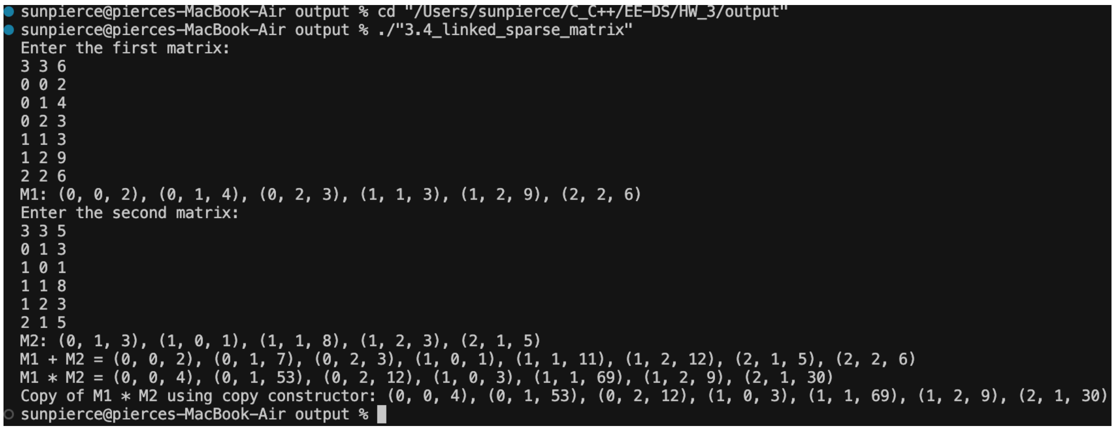
(c) Write the C++ function, **operator**<<(), which outputs a sparse matrix as triples (i, j, a_{ij}).

(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.

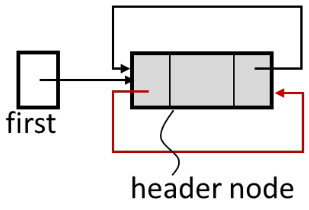
```
sol:
Execution trace:
```



```
sunpierce@pierces-MacBook-Air output % cd "/Users/sunpierce/C_C++/EE-DS/HW_3/output"
sunpierce@pierces-MacBook-Air output % ./"3.4_linked_sparse_matrix"
Enter the first matrix:
3 3 6
0 0 2
0 1 4
0 2 3
1 1 3
1 2 9
2 2 6
M1: (0, 0, 2), (0, 1, 4), (0, 2, 3), (1, 1, 3), (1, 2, 9), (2, 2, 6)
Enter the second matrix:
3 3 5
0 1 3
1 0 1
1 1 8
1 2 3
2 1 5
M2: (0, 1, 3), (1, 0, 1), (1, 1, 8), (1, 2, 3), (2, 1, 5)
M1 + M2 = (0, 0, 2), (0, 1, 7), (0, 2, 3), (1, 0, 1), (1, 1, 11), (1, 2, 12), (2, 1, 5), (2, 2, 6)
M1 * M2 = (0, 0, 4), (0, 1, 53), (0, 2, 12), (1, 0, 3), (1, 1, 69), (1, 2, 9), (2, 1, 30)
Copy of M1 * M2 using copy constructor: (0, 0, 4), (0, 1, 53), (0, 2, 12), (1, 0, 3), (1, 1, 69), (1, 2, 9), (2, 1, 30)
sunpierce@pierces-MacBook-Air output % 
```

5. (10%) doubly linked circular list

Implement the template doubly linked circular list with header node DblList ADT in textbook.



```
class DblList;    //forward declaration
class DblListNode {
friend class DblList;
private:
    int data;
    DblListNode * left, * right;
};
class DblList {
public:
   // List manipuation operations
   DblList();
   ~DblList();.
   void Insert(DblListNode *p, DblListNode *x);
   void Delete(DblListNode *x);
   .
private:
    DblListNode *head;    // points to header node
};
```

You should

(a) Implement the ADT fully (including destructor and necessary constructors)

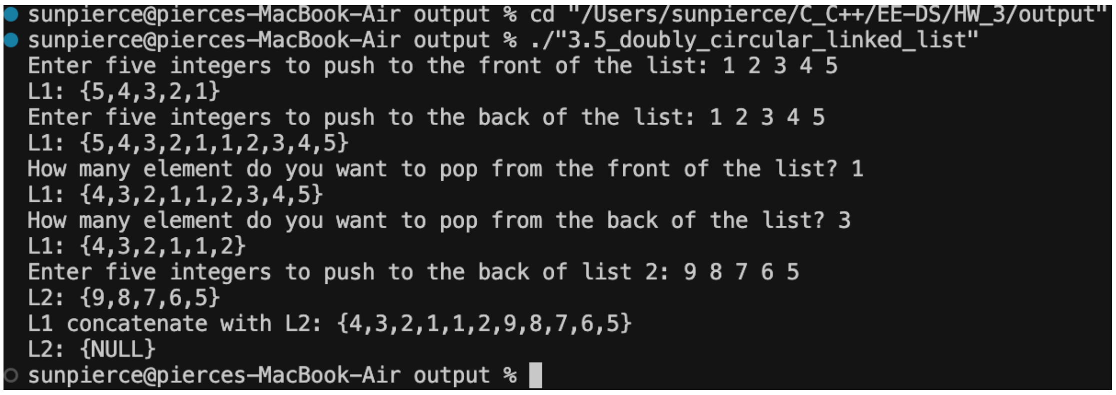
(b) Add void Concatenate(DblList m) to concatenate the two lists *this and m. On completion of the function, the resulting list should be stored in *this and the list m should contain the empty list. Your function must run in O(1) time.

(c) Add Functions, Push(x), Pop, Inject(x), Eject(), to insert and delete at either end of the list in O(1) time. (Such functions are needed for a structure called a deque.)

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

sol:

Execution trace:



```
sunpierce@pierces-MacBook-Air output % cd "/Users/sunpierce/C_C++/EE-DS/HW_3/output"
sunpierce@pierces-MacBook-Air output % ./"3.5_doubly_circular_linked_list"
Enter five integers to push to the front of the list: 1 2 3 4 5
L1: {5,4,3,2,1}
Enter five integers to push to the back of the list: 1 2 3 4 5
L1: {5,4,3,2,1,1,2,3,4,5}
How many element do you want to pop from the front of the list? 1
L1: {4,3,2,1,1,2,3,4,5}
How many element do you want to pop from the back of the list? 3
L1: {4,3,2,1,1,2}
Enter five integers to push to the back of list 2: 9 8 7 6 5
L2: {9,8,7,6,5}
L1 concatenate with L2: {4,3,2,1,1,2,9,8,7,6,5}
L2: {NULL}
sunpierce@pierces-MacBook-Air output %
```