

Data Structure Program Assignment #6  
(Due: PM:6:00, May 20, 2025)  
**Polynomial operations by template linked list**  
*Instructor: Jiann-Jone Chen*

**I. Introduction**

1. This assignment is designed to demonstrate the implementation of a polynomial class using a linked list structure.

In this program, the traditional linked list and list node structures have been updated to **Chain** and **ChainNode**, respectively, in accordance with the latest textbook guidelines. While the data structures have been modernized, the core functions and underlying algorithms remain unchanged.

2. During the lecture, the process of adding two polynomials represented by a template-based linked list (**Chain**) was demonstrated. In this assignment, you must implement both polynomial addition and multiplication operations based on this updated template-linked list structure.
3. Developing a strong understanding of linked list design is a fundamental skill in C++ programming. You are strongly encouraged to design and complete this program independently, if possible, to strengthen your skills.
4. In Homework #3, you previously designed algorithms and programs to perform polynomial addition and multiplication operations.

For this assignment, the algorithms themselves remain the same; however, you must adapt your implementation to use the redesigned linked list structure introduced in Lecture 4 materials.

**II. Steps:**

1. The algorithm to perform polynomial addition has been introduced in lecture notes Linked List I-47.
2. You can design the (a) template ListNode class; (b) template List class; and (c) ListIterator class; and (d) polynomial class, respectively, according to the programs/algorithms shown in lecture notes. For example, ListNode and List (lecture notes I 11, 15), ListIterator(lecture notes I-22~25), polynomial (lecture notes I 45, 47, 49).
3. The main program is shown below:

```

#include "ChainNode.h"
#include "Polynomial.h"
#include "av_list.h"
#include "gui.h"
#include <locale>
#include <codecvt> // for wstring <-> string conversion
#include <fstream>

int debug = 0;
int N;
extern void LOGO();
void system_init(std::ifstream&);

// New function: handle file dialog + open file
std::ifstream openInputFile();

int main() {
    int cont = 1;
    char ch;

    while (cont) {
        // Use the new function to open the input file
        std::ifstream inFile = openInputFile();
        if (!inFile) {
            std::cerr << "Program terminated due to file opening failure.\n";
            return 0;
        }
        Polynomial a, b, c, d;
        LOGO();
        system_init(inFile);

        // Read two polynomials
        inFile >> a;
        inFile >> b;

        // Polynomial addition and multiplication
        c = (a + b);
        std::cout << "a = " << a << "\nb = " << b << "\nc = a + b =\n"
            << a << " + " << b << " =\n " << c << "\n";

        d = a * b;
        d = c + d;
        cout << "\n(a+b) + (a*b) =\n" << c << " + " << a << " * " << b << " =\n" << d << "\n";

        cont = 0;

        // Trigger Polynomial destructors and free all nodes
        std::cout << "Press any key to trigger Polynomial destructors and release
memory...\n";
        ch = getchar();
    }

    std::cout << "Press any key to exit the program...\n";
    getchar();
    return 0;
}

```

#### 4. The struct Term and Polynomial classes are shown below

```

struct Term { // all members of Term are public by default
    int coef; //coefficient
    int exp; // exponential lorder
    Term Set(int c, int e) {
        coef = c; exp = e; return *this; };
    Term Set(ChainNode <Term> temp) {
        Term data = temp.GetData();
        coef = data.coef; exp = data.exp;
        return *this;
    };
};

class Polynomial {
    friend ostream &operator<<(ostream &, Polynomial &);
    friend istream &operator>>(istream &, Polynomial &);
    friend Polynomial operator+(const Polynomial&, const Polynomial&);

public:
    Polynomial(){ label = 'A' + counter; num = counter++; message(); };
    Polynomial(char c) { label = c; num = counter++; message(); };
    Polynomial(const Polynomial&); // copy constructor
    ~Polynomial();
    void operator=(const Polynomial&);
    Polynomial operator*(const Polynomial&) const;

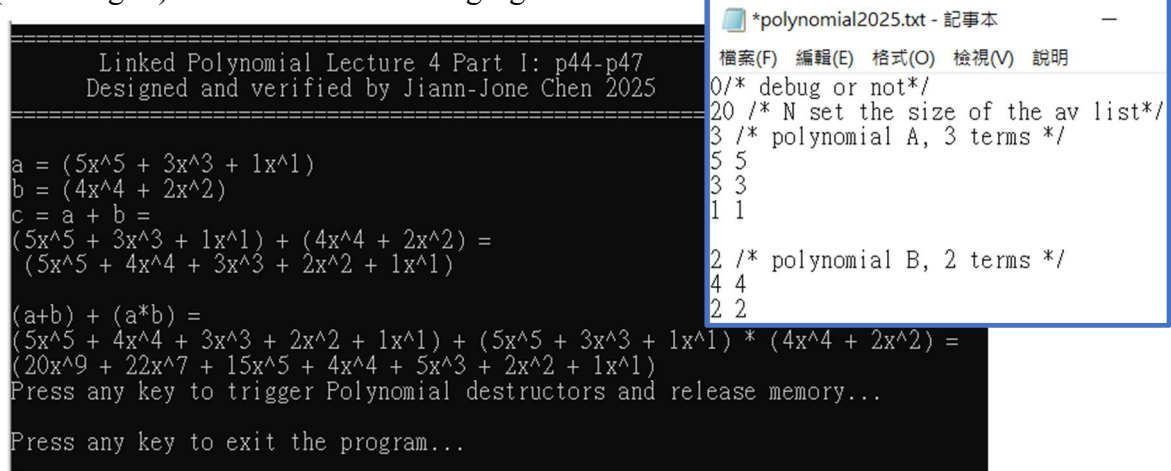
public:
    Chain <Term> poly;
    char label;

private:
    void message();
    static int counter;
    int num;
};

```

#### 5. Results:

When the polynomial2025.txt is set to the one on the right. The correct execution result (set debug=0) looks like the following figure:



The image shows two windows. The main window is a terminal with a black background and white text, displaying the output of a polynomial program. It shows the creation of two polynomials, A and B, and their addition and multiplication. The input file 'polynomial2025.txt' is shown in a separate window on the right, containing comments and numerical data for the polynomials.

```

Linked Polynomial Lecture 4 Part I: p44-p47
Designed and verified by Jiann-Jone Chen 2025

a = (5x^5 + 3x^3 + 1x^1)
b = (4x^4 + 2x^2)
c = a + b =
(5x^5 + 3x^3 + 1x^1) + (4x^4 + 2x^2) =
(5x^5 + 4x^4 + 3x^3 + 2x^2 + 1x^1)

(a+b) + (a*b) =
(5x^5 + 4x^4 + 3x^3 + 2x^2 + 1x^1) + (5x^5 + 3x^3 + 1x^1) * (4x^4 + 2x^2) =
(20x^9 + 22x^7 + 15x^5 + 4x^4 + 5x^3 + 2x^2 + 1x^1)
Press any key to trigger Polynomial destructors and release memory...

Press any key to exit the program...

```

The input file 'polynomial2025.txt' contains the following content:

```

/* debug or not*/
20 /* N set the size of the av list*/
3 /* polynomial A, 3 terms */
5 5
3 3
1 1

2 /* polynomial B, 2 terms */
4 4
2 2

```

When we set debug=1, the execution result had to show the memory allocation details when your program uses the “new” function to allocate memory for objects.

```
new node( 5, 5)->Initial Total AV nodes =20
```

```
total AV nodes = 19
new node( 3, 3)->total AV nodes = 18
new node( 1, 1)->total AV nodes = 17
new node( 4, 4)->total AV nodes = 16
new node( 2, 2)->total AV nodes = 15
Construct Poly(num=4, op=+)
new node( 5, 5)->total AV nodes = 14
new node( 4, 4)->total AV nodes = 13
new node( 3, 3)->total AV nodes = 12
new node( 2, 2)->total AV nodes = 11
new node( 1, 1)->total AV nodes = 10
new node( 5, 5)->total AV nodes = 9
new node( 4, 4)->total AV nodes = 8
new node( 3, 3)->total AV nodes = 7
new node( 2, 2)->total AV nodes = 6
new node( 1, 1)->total AV nodes = 5
Destruct Poly(num=4, op= +): Delete
recycle 5 nodes->total AV nodes = 10
a = (5x^5 + 3x^3 + 1x^1)
b = (4x^4 + 2x^2)
c = a + b =
(5x^5 + 3x^3 + 1x^1) + (4x^4 + 2x^2) =
(5x^5 + 4x^4 + 3x^3 + 2x^2 + 1x^1)
Construct Poly(num=5, op=*)
User Defined :: Operator newl6
new node(20, 9)->total AV nodes = 9
new node(10, 7)->total AV nodes = 8
User Defined :: Operator delete
User Defined :: Operator newl6
new node(12, 7)->total AV nodes = 7
recycle node(12,7)->total AV nodes = 8
new node( 6, 5)->total AV nodes = 7
User Defined :: Operator delete
User Defined :: Operator newl6
new node( 4, 5)->total AV nodes = 6
recycle node(4,5)->total AV nodes = 7
new node( 2, 3)->total AV nodes = 6
User Defined :: Operator delete
User Defined :: Operator newl6
new node(20, 9)->total AV nodes = 5
new node(22, 7)->total AV nodes = 4
new node(10, 5)->total AV nodes = 3
new node( 2, 3)->total AV nodes = 2
Destruct Poly(num=5, op= *): Delete
recycle 4 nodes->total AV nodes = 6
```

```
polynomial.txt - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明
1| /* debug =1 for debugging */
20| /* N sets the size of the av */
2| /* polynomial A */
3| 3
1| 1
2| /* polynomial B */
4| 4
2| 2
```

```

Construct Poly(num=6, op=+)
new node(20, 9)->total AV nodes = 5
new node(22, 7)->total AV nodes = 4
new node(15, 5)->total AV nodes = 3
new node( 4, 4)->total AV nodes = 2
new node( 5, 3)->total AV nodes = 1
new node( 2, 2)->total AV nodes = 0
new node( 1, 1)->Initial Total AV nodes =20
total AV nodes = 19
recycle 0 nodes->total AV nodes = 23

new node(20, 9)->total AV nodes = 22
new node(22, 7)->total AV nodes = 21
new node(15, 5)->total AV nodes = 20
new node( 4, 4)->total AV nodes = 19
new node( 5, 3)->total AV nodes = 18
new node( 2, 2)->total AV nodes = 17
new node( 1, 1)->total AV nodes = 16
Destruct Poly(num=6, op= +): Delete
recycle 7 nodes->total AV nodes = 23

(a+b) + (a*b) =
(5x^5 + 4x^4 + 3x^3 + 2x^2 + 1x^1) + (5x^5 + 3x^3 + 1x^1) * (4x^4 + 2x^2) =
(20x^9 + 22x^7 + 15x^5 + 4x^4 + 5x^3 + 2x^2 + 1x^1)
Press any key to trigger Polynomial destructors and release memory...

Destruct Poly(num=3, op= D): Delete
recycle 7 nodes->total AV nodes = 30

Destruct Poly(num=2, op= C): Delete
recycle 5 nodes->total AV nodes = 35

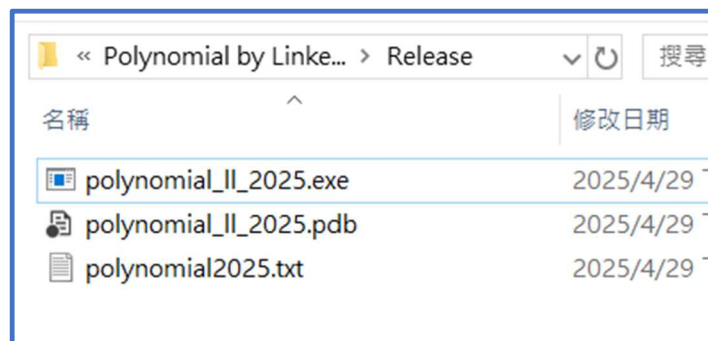
Destruct Poly(num=1, op= B): Delete
recycle 2 nodes->total AV nodes = 37

Destruct Poly(num=0, op= A): Delete
recycle 3 nodes->total AV nodes = 40
Press any key to exit the program...

```

The av list was initialized two times and it finally comprises 40 nodes.

A demo program, **polynomial\_II\_2025.exe**, and a file, **polynomial2025.txt**, are provided for you to verify your program's correctness.



A demo project is provided for you to focus on designing major algorithms.

### III. Requirements:

1. You have to submit the compressed files of the complete project, named with your student ID, such that the TA can recompile your programs to test correctness.
2. You have to write a short report to describe
  - (a) What is the program all about?
  - (b) What functions have you designed to provide a calculator program?
  - (c) What and how to solve problems during the program design.
  - (d) How will you improve this program?

### IV. Criteria

1. (85%) Implement the polynomial class and perform addition and multiplication based on the template linked list class.  
i.e., your program can output the correct results.
2. (15%) Implement the memory allocation process based on an available list that can pop and insert nodes to its list to serve as new and delete functions for the entire program.

### V. Hint

1. You can write your own function to new and delete node data from the available list. For example, implement the Getnode and the Retnode member functions shown in linked list I-53 to serve new and delete functions.
2. You can also try to design the template listnode and list classes with function overloading to manage the memory allocation for new and delete functions. That is, both the new and delete commands are reserved.



A template ChainNode class and a template chain class are provided for your reference. You can overload the new and delete functions for memory allocation operations. (Hint: use a **static** member class to deal with an available list.)

```
template <class T>
class ChainNode {
    friend class Polynomial;
    friend class Chain <T>;
    friend class ChainIterator <T>; // new added for iterator
    friend istream &operator>>(istream &, Polynomial&);
    template <class T> friend ostream &operator<<(ostream &, Chain<T>&);
public:
    T GetData() { return data; };
    ChainNode <T>* GetLink() { return link; };
    ChainNode(T element, ChainNode<T>* l = 0) :data(element), link(l) { };
    void* operator new(size_t size) { // overloading new
        void* p; /* get a node from the av list pointed by p */
        return p;
    }
    void operator delete(void* p) { // overloading delete
        /* return the node data pointed by p to the av list */
    }
private:
    T data;
    ChainNode<T> *link;
};
```

```
// *****Chain Template Class *****
template<class T>
class Chain {
    template <class T> friend ostream& operator<<(ostream&, Chain<T>&);
    friend class ChainIterator<T>; // New Added for Iterator
    friend class Polynomial;
    friend class ChainNode<T>;
    // friend void *av_init(void*, int);
    friend istream& operator >> (istream&, Polynomial&); // read in a matrix
public:
    Chain() { current = first = last = NULL; num = 0; }; // constructor
    ~Chain(); // destructor to free all allocated memory space, got some trouble
    void Create2(T, T);
    void Insert(ChainNode<T>*, T);
    ChainNode<T>* Pop();
    void Delete(ChainNode<T>*, ChainNode<T>*);
    void InsertBack(const T& e);
    void InsertBack(ChainNode<T>*);
    void InsertInorder(const T& e);
    ChainNode<T>* FirstNode() { return first; };
    void PrintChain();
    void operator=(const Chain<T>& t);
private:
    ChainNode <T>* first, * last, *current;
    int num;
};
```