哈尔滨工业大学计算机科学与技术学院

实验报告

课程名称： 数据结构与算法设计

课程类型：必修

实验项目名称： Polynomials

实验题目： 多项式 ADT 的实现

班级：1503101
学号：1150310116
姓名：李博

设计成绩	报告成绩	指导老师

一、实验目的

使用链表存储稀疏多项式并实现一元多项式的代数运算，熟悉链表相关的插入删除操作，掌握内存池，指针管理等相关知识。

二、实验要求及实验环境

1.能够输入多项式(可以按各项的任意输入顺序，建立按指数降幂排列的多项式和输出多项式(按指数降幂排列)

2.能够计算多项式在某一点 x=x0 的值，其中 x0 是一个浮点型常量，返回结果，为浮点数。

3.能够给出计算两个多项式加法、减法、乘法和除法运算的结果多项式，除法运算的结果包括商多项式和余数多项式。

4.利用循环链表结构和可用空间表的思想，把循环链表表示的多项式返还给可用空间表，以减少乘法和除法运算中间结果的空间占用和结点频繁的分配与回收操作。

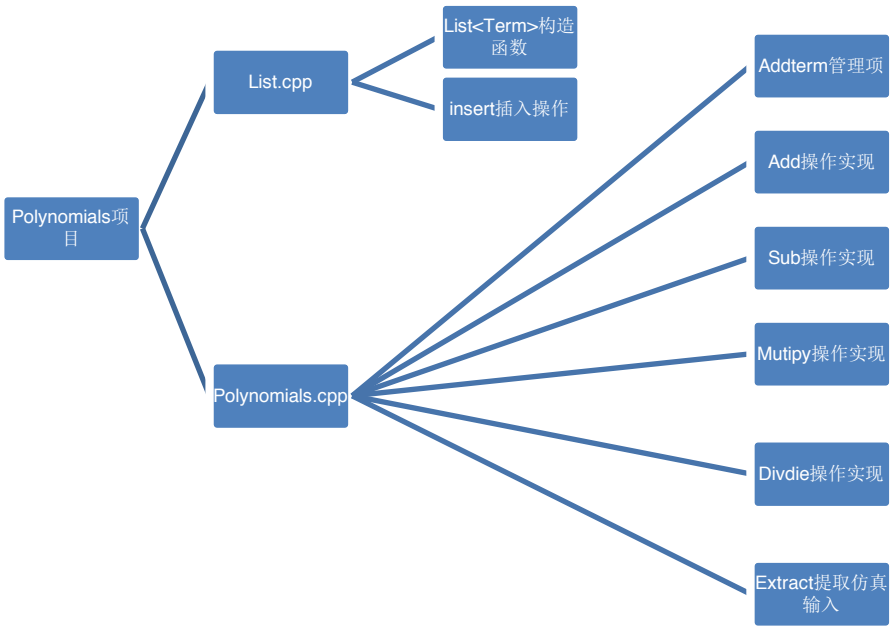
三、设计思想（本程序中的用到的所有数据类型的定义，主程序的流程图及各程序模块之间的调用关系）

1．逻辑设计

1.此次实验中利用了一个自己以前设计的 List 的类，容器内的元素为新定义 Term 结构。

2.Term 结构中包含了 double 类型的系数，int 类型的指数，以及重定义了<运算，在测试时我使用 vector 容器可以将其按照指数从大到小排序，但实际代码中并未使用到这个<运算符，使用了一个 addterm 函数从前往后找，然后插入结点，始终维护前面项目的指数最大。

3.在 Polynomial 文件中设计了有关于多项式的加减乘除，求值的函数，具体解释如下。



2.方法解释

- List 的构造函数可以适用于各种类型，因此我使用了一个 Term 类型的元素，使用 List<Term>操作初始化一个链表，和 STL 使用方法类似，将 List 和多项式操作完全分离开。

- Insert 操作在 Addterm 里面大量的使用，时间复杂度 O（1）；
- AddTerm 操作：始终维护链表中元素的 Exp 从大到小排序，每次插入都从前往后找即可。
- Add 操作：对于两个链表，分别定义 iterator 遍历，指数相同则合并，否则加入指数较大的那一个元素，然后 iterator 后移。
- Sub 操作：将减数链表取负，执行 Add 操作即可。
- Multiply 操作：对 B 链表中的每一个元素都乘上 A 链表，然后合并同类项，可以使用循环链表实现，对 B1 使用完乘法之后，直接 B2 对应 A 的开头。
- Divide 操作：每次对于 A 链表的最大项，试除 B 链表的最大项，直到 A 的最大项小于 B 的最大项结束条件。在条件中，每次将商加入成新链表，然后用商乘上除数 B，再用 A = A – B 即可。
  余数多项式 =A – B * result
- Extract 操作：以上的操作都很幼稚，只有这个操作比较的费时间，但是实现起来比较有意义，可以复用。这个函数可以将输入的 32.2x^4+2x^3 这样的 string 类，其中的系数和指数提取出来，使输入更加的仿真。具体就是根据+号将原串分成若干个子串，再对子串以 x^为界限，分别提取系数和指数，其中有许多特例，如系数为 1，指数为 1，没有 x 项等等，但是理清楚所有情况，分类之后写成一个 if-else 树也可以很好的实现。

四、测试结果

1.测试输入信息
add
x^23+12x^3+343x^32
x^14+32x^2+2

sub
x^23+12x^3+343x^32
x^14+32x^2+2

div
x^23+12x^3+343x^32
x^14+32x^2+2

mut
x^23+12x^3+343x^32
x^14+32x^2+2

get
x^23+12x^3+343x^32

1.132

2.测试输出信息

Welcome to calculator of polynomials
Enter Add To Get The Sum Of Two Polynomials
Enter Sub To Subtract Two Polynomials

Enter Mut To Mutiply Two Polynomials
Enter Div To Divide A from B
Enter Mod To Mod A from B
Enter Get To Get Value In A Specified Polynomial
Enter Other Button To Quit
What's your choice :
Please Input The Polynomials: You Have Inserted A Polynomials : 343x^32 + 1x^23 + 12x^3

Please Input The Polynomials: You Have Inserted A Polynomials : 1x^14 + 32x^2

The Ans of A + B = 343x^32 + 1x^23 + 1x^14 + 12x^3 + 32x^2

Welcome to calculator of polynomials
Enter Add To Get The Sum Of Two Polynomials
Enter Sub To Subtract Two Polynomials
Enter Mut To Mutiply Two Polynomials
Enter Div To Divide A from B
Enter Mod To Mod A from B
Enter Get To Get Value In A Specified Polynomial
Enter Other Button To Quit
What's your choice :
Please Input The Polynomials: You Have Inserted A Polynomials : 343x^32 + 1x^23 + 12x^3

Please Input The Polynomials: You Have Inserted A Polynomials : 1x^14 + 32x^2

The Ans of A - B = 343x^32 + 1x^23 + -1x^14 + 12x^3 + -32x^2

Welcome to calculator of polynomials
Enter Add To Get The Sum Of Two Polynomials
Enter Sub To Subtract Two Polynomials
Enter Mut To Mutiply Two Polynomials
Enter Div To Divide A from B
Enter Mod To Mod A from B
Enter Get To Get Value In A Specified Polynomial
Enter Other Button To Quit
What's your choice :
Please Input The Polynomials: You Have Inserted A Polynomials : 343x^32 + 1x^23 + 12x^3

Please Input The Polynomials: You Have Inserted A Polynomials : 1x^14 + 32x^2

The Ans of A / B = 343x^18 + 1x^9 + -10976x^6
The Remainder Polynomial = -32x^11 + 351232x^8 + 12x^3

Welcome to calculator of polynomials
Enter Add To Get The Sum Of Two Polynomials
Enter Sub To Subtract Two Polynomials
Enter Mut To Mutiply Two Polynomials
Enter Div To Divide A from B
Enter Mod To Mod A from B

Enter Get To Get Value In A Specified Polynomial
Enter Other Button To Quit
What's your choice :
Please Input The Polynomials: You Have Inserted A Polynomials : 343x^32 + 1x^23 + 12x^3

Please Input The Polynomials: You Have Inserted A Polynomials : 1x^14 + 32x^2

The Ans of A * B = 343x^46 + 1x^37 + 10976x^34 + 32x^25 + 12x^17 + 384x^5

Welcome to calculator of polynomials
Enter Add To Get The Sum Of Two Polynomials
Enter Sub To Subtract Two Polynomials
Enter Mut To Mutiply Two Polynomials
Enter Div To Divide A from B
Enter Mod To Mod A from B
Enter Get To Get Value In A Specified Polynomial
Enter Other Button To Quit
What's your choice :
Please Input The Polynoimal That You Want To Caculate
Please Input The Polynomials: You Have Inserted A Polynomials : 343x^32 + 1x^23 + 12x^3

Please Input The X0 That You Want To Get: The Value Is: 18164

Welcome to calculator of polynomials
Enter Add To Get The Sum Of Two Polynomials
Enter Sub To Subtract Two Polynomials
Enter Mut To Mutiply Two Polynomials
Enter Div To Divide A from B
Enter Mod To Mod A from B
Enter Get To Get Value In A Specified Polynomial
Enter Other Button To Quit
What's your choice :

五、系统不足与经验体会

系统不足：想在 addterm 函数中使用一个最大堆来管理 Polynomial 类中的项实现从大到小排列，这样插入可以实现 log(n)的操作想实现一些极端情况下的异常处理，另外这个程序在 Xcode 下模拟大数据输入会崩掉 IDE，所以有可能其在内存的使用上可能还存在一些问题（但使用 Sublime 配合 command，打开并未出现这种情况所以也有可能是 Xcode 的问题）

心得体会：这个实验本来是打算写成一个可以自己使用的完备的 Polynomials 类，所以开始估计 5 小时的时间来完成，但是最后多花了三个小时在处理多项式的仿真输入时（就是读入 x^3 + 3.2x^5 这样的 string，然后再由函数进行相应的处理)，又加上这段时间比较的忙，所以时间上很紧张。因此对于这次实验的这种比较简单的代码，以

后还是应该尽快的完成一个能够使用的系统，然后再考虑在这个系统上逐步求精，直到最后写出一个完整的工程类。

六、附录：源代码（带注释）

//List.cpp

```
#include <iostream>
#include <fstream>
//可以相应的增加First(),End()方法，表示表头和表尾。

using namespace std;

template <typename Object>
class List
{
private:
	struct Node
	{
		Object data;
		Node *prev;
		Node *next;
		//引用了object的构造函数
		Node (const Object &d = Object(), Node *p = NULL,Node *n = NULL)
		: data(d),prev(p),next(n)
		{

		}
	};

public:
	class const_iterator
	{
	public:
		const_iterator():current(NULL)
		{

		}
		const Object &operator *() const
		{
			return retrieve();
		}
		//++itr的实现
		const_iterator &operator ++()
		{
			current = current->next;
			return *this;
		}
```

```
    //itr++的实现
    const_iterator operator ++ (int)
    {
        const_iterator old = *this;
        ++(*this);
        return old;
    }
    bool operator == (const const_iterator &rhs) const
    {
        return current == rhs.current;
    }
    bool operator != (const const_iterator &rhs) const
    {
        return !(*this == rhs);
    }
protected:
    //iterator里面包含了指向当前节点的元素指针
    Node *current;
    Object & retrieve() const
    {
        return current->data;
    }
    const_iterator (Node *p) : current(p)
    {

    }
    friend class List<Object>;
};
class iterator : public const_iterator
{
public:
    iterator()
    {

    }
    Object & operator *()
    {
        return const_iterator::retrieve();
    }
    const Object &operator *() const
    {
        return const_iterator::operator *();
    }
    iterator & operator ++()
    {
        const_iterator :: current = const_iterator :: current->next;
        return *this;
    }
    iterator operator ++(int)
    {
        iterator old = *this;
```

```
            ++(*this);
            return old;
        }
    protected:
        iterator(Node *p) : const_iterator(p)
        {

        }
        friend class List<Object>;
    };
public:
    List()
    {
        init();
    }
    List(const List & rhs)
    {
        init();
        *this = rhs;
    }
    ~List()
    {
        clear();
        delete head;
        delete tail;
    }
    const List &operator = (const List &rhs)
    {
        if (this == &rhs)
        {
            return *this;
        }
        clear();
        for (const_iterator itr = rhs.begin(); itr != rhs.end(); ++itr)
        {
            push_back(*itr);
        }
        return *this;
    }
    iterator begin()
    {
        return iterator (head->next);
    }
    const_iterator begin() const
    {
        return iterator(head->next);
    }
    iterator end()
    {
        return tail;
    }
    const_iterator end() const
```

```
{
    return const_iterator(tail);
}
int size() const
{
    return theSize;
}
bool empty() const
{
    return size() == 0;
}
void clear()
{
    while (!empty())
    {
        pop_front();
    }
}
Object &front()
{
    return *begin();
}
const Object &front() const
{
    return *begin();
}
Object & back()
{
    return *(--end());
}
const Object & back() const
{
    return *(--end());
}
void push_front(const Object &x)
{
    insert(begin(),x);
}
void push_back(const Object &x)
{
    insert(end(),x);
}
void pop_front()
{
    erase(begin());
}
void pop_back()
{
    erase(--end());
}
iterator insert(iterator itr,const Object &x)
{
```

```
            Node *p = itr.current;
            theSize ++;
            return iterator(p->prev = p->prev->next = new Node(x,p->prev,p));
        }
        iterator erase(iterator itr)
        {
            Node *p = itr.current;
            iterator retVal(p->next);
            p->prev->next = p->next;
            p->next->prev = p->prev;
            delete p;
            theSize --;
            return retVal;
        }
private:
        int theSize;
        Node *head;
        Node *tail;
        void init()
        {
            theSize = 0;
            head = new Node;
            tail = new Node;
            head->next = tail;
            tail->prev = head;
        }
};
```

//Polynomial.cpp

```
//
//  main.cpp
//  Polynomials
//
//  Created by 李博 on 2016/10/30.
//  Copyright © 2016年 李博. All rights reserved.
//

#include <iostream>
#include <fstream>
#include <sstream>
#include <algorithm>
#include <vector>
#include <stack>
#include <cmath>
#include "List.cpp"

using namespace std;
```

```
struct Term
{
    int Exp;
    double Coef;
    bool operator < (const Term &rhs) const
    {
        return Exp > rhs.Exp;
    }
    Term(double a = 0,int b = 0)
    {
        Exp = b;
        Coef = a;
    }
};

string StringToLower(string &name)
{
    for (int i = 0; i < name.size(); ++i)
    {
        if (isupper(name[i]))
        {
            name[i] = tolower(name[i]);
        }
    }
    return name;
}

int ExecuteString(const string &temp,int index,bool forward)
{
    int accumu = 0;
    stack<int> Contain;
    while (!Contain.empty())
    {
        Contain.pop();
    }
    if (forward == 1)
    {
        for (int i = index; i < temp.size() && (isdigit(temp[i])); ++i)
        {
            accumu += temp[i] - '0';
            accumu *= 10;
        }
    }
    else
    {
        for (int i = index; i >= 0 && (isdigit(temp[i])); -- i)
        {
            Contain.push(temp[i] - '0');
        }
        while (!Contain.empty())
        {
```

```
            int cache = Contain.top();
            accumu += (cache);
            Contain.pop();
            accumu *= 10;
        }
    }
    return (accumu / 10);
}

// queue <Term *> freenodes;
// Node Term[2000];
// void init()
// {
//    for (int i = 0; i < 2000; ++i)
//    {
//        freenodes.push(&Term[i]);
//    }
// }

// Node* newnode()
// {
//    Node *u = freenodes.front();
//    freenodes.pop();
//    return u;
// }

// Node* deletenode(Node *u)
// {
//    freenodes.push(u);
// }

void addterm(List<Term> &a,Term &b)
{
    List<Term> :: iterator itr;
    for (itr = a.begin(); itr != a.end(); ++itr)
    {
        if ((*itr).Exp > b.Exp)
        {
            continue;
        }
        else
        {
            break;
        }
    }
    if (itr == a.end())
    {
        a.insert(itr,b);
    }
    else if ((*itr).Exp == b.Exp)
    {
        (*itr).Coef += b.Coef;
```

```
    }
    else
    {
        a.insert(itr,b);
    }
}

List<Term> PolynomialsAdd(List<Term> &a,List<Term> &b)
{
    List<Term> :: iterator at = a.begin();
    List<Term> :: iterator bt = b.begin();
    Term newTerm;
    List<Term> Sum;
    while (at != a.end() && bt != b.end())
    {
        if ((*at).Exp == (*bt).Exp)
        {
            newTerm.Coef = (*at).Coef + (*bt).Coef;
            newTerm.Exp = (*at).Exp;
            Sum.push_back(newTerm);
            ++at;
            ++bt;
        }
        else if ((*at).Exp > (*bt).Exp)
        {
            newTerm.Coef = (*at).Coef;
            newTerm.Exp = (*at).Exp;
            Sum.push_back(newTerm);
            ++at;
        }
        else if((*at).Exp < (*bt).Exp)
        {
            newTerm.Coef = (*bt).Coef;
            newTerm.Exp = (*bt).Exp;
            Sum.push_back(newTerm);
            ++bt;
        }
    }
    while (at != a.end())
    {
        newTerm.Coef = (*at).Coef;
        newTerm.Exp = (*at).Exp;
        Sum.push_back(newTerm);
        ++at;
    }
    while (bt != b.end())
    {
        newTerm.Coef = (*bt).Coef;
        newTerm.Exp = (*bt).Exp;
        Sum.push_back(newTerm);
        ++bt;
    }
```

```
    return Sum;
}

//define that a - b
List<Term> PolynomialsSub(List<Term> &a,List<Term> &b)
{
    List<Term> :: iterator bt;
    for (bt = b.begin(); bt != b.end(); ++bt)
    {
        (*bt).Coef = -(*bt).Coef;
    }
    return PolynomialsAdd(a,b);
}

//store in Polynomials[2]
//a * b
List<Term> PolynomialsMutipy(List<Term> &a,List<Term> &b)
{
    List<Term>::iterator at;
    List<Term>::iterator bt;
    Term newTerm;
    List<Term> Ans;
    List<Term> Temp;
    Ans.clear();
    for (at = a.begin(); at != a.end(); ++at)
    {
        Temp.clear();
        for (bt = b.begin(); bt != b.end(); ++bt)
        {
            double CoefAns = (*at).Coef * (*bt).Coef;
            int ExpAns = (*at).Exp + (*bt).Exp;
            newTerm.Coef = CoefAns;
            newTerm.Exp = ExpAns;
            addterm(Temp,newTerm);
        }
        Ans = PolynomialsAdd(Temp, Ans);
    }
    return Ans;
}

void Display(List<Term> &a)
{
    if (a.size() == 0)
    {
        cout<<0;
    }
    List<Term> :: iterator itr;
    int countsize = 0;
    for (itr = a.begin(); itr != a.end(); ++itr)
    {
        countsize ++;
        if ((*itr).Coef == 0)
```

```
        {
            continue;
        }
        else if ((*itr).Exp == 0)
        {
            cout<<(*itr).Coef;
        }
        else
        {
            cout<<(*itr).Coef<<"x^"<<(*itr).Exp;
        }
        if(countsize != a.size())
        {
            cout<<" + ";
        }
    }
    cout<<"\n";
}

double CaculateValue(List<Term> &a,double x)
{
    double Ans = 0;
    List<Term> :: iterator itr;
    for (itr = a.begin(); itr != a.end(); ++itr)
    {
        Ans += (*itr).Coef * (pow(x,(*itr).Exp));
    }
    return Ans;
}

void check_poly(List<Term> &p1)
{
    for (List<Term> :: iterator itr = p1.begin(); itr != p1.end(); ++itr)
    {
        if ((*itr).Coef == 0)
        {
            p1.erase(itr);
        }
    }
}

List<Term> try_divide(List<Term> &p1,List<Term> &p2)
{
    List<Term> quotient;
    List<Term> remainder;
    while ((*(p1.begin())).Exp >= (*(p2.begin())).Exp)
    {
        double temp_coef = (*(p1.begin())).Coef / (*(p2.begin())).Coef;
        int temp_exp = (*(p1.begin())).Exp - (*(p2.begin())).Exp;
        List<Term> temp;
        temp.clear();
        Term newTerm(temp_coef,temp_exp);
```

```
        temp.push_back(newTerm);
        addterm(quotient,newTerm);
        temp = PolynomialsMutipy(temp,p2);
        p1 = PolynomialsSub(p1,temp);
        for (List<Term> :: iterator itr = p1.begin(); itr != p1.end(); ++itr)
        {
            if ((*itr).Coef == 0)
            {
                p1.erase(itr);
            }
        }
    }
    return quotient;
}

List<Term> try_mod(List<Term> &p1,List<Term> &p2)
{
    List<Term> remainder = p1;
    List<Term> temp;
    temp = try_divide(p1, p2);
    check_poly(temp);
    temp = PolynomialsMutipy(temp, p2);
    remainder = PolynomialsSub(remainder, temp);
    check_poly(remainder);
    return remainder;
}

List<Term> Extract(const string &a)
{
    List<Term> ret;
    vector<int> StorePlus;
    for (int i = 0; i < a.size(); ++i)
    {
        if (a[i] == '+')
        {
            StorePlus.push_back(i);
        }
    }
    Term newTerm;
    double Coef = 0;
    int Exp = 0;
    stringstream transfer;
    transfer.clear();
    string substring;
    int prev = 0;
    int idx = 0;
    for (int i = 0; i < StorePlus.size(); ++i)
    {
        transfer.clear();
        bool FindX = 0;
        for (int j = prev; j < StorePlus[i]; ++j)
        {
```

```
    if (a[j] == 'x')
    {
        FindX = 1;
        idx = j;
        break;
    }
}
if (FindX)
{
    if (idx == 0)
    {
        Coef = 1;
        if(a[idx+1] == '^')
        {
            if(i + 1 < StorePlus.size())
            {
                substring = a.substr(idx+2,StorePlus[i]-idx-2);
            }
            else
            {
                substring = a.substr(idx + 2,a.size()-idx - 2);
            }
            transfer.clear();
            transfer<<substring;
            transfer>>Exp;
        }
        else
        {
            Exp = 1;
        }
    }
    else
    {
        if(a[idx-1] == '+')
        {
            Coef = 1;
        }
        else
        {
            substring = a.substr(StorePlus[i-1]+1,idx - StorePlus[i-1] - 1);
            transfer.clear();
            transfer<<substring;
            transfer>>Coef;
        }

        if(a[idx+1] == '^')
        {
            if(i + 1 < StorePlus.size())
            {
                substring = a.substr(idx+2,StorePlus[i]-idx-2);
            }
            else
```

```
            {
                substring = a.substr(idx + 2,a.size()-idx - 2);
            }
            transfer.clear();
            transfer<<substring;
            transfer>>Exp;
        }
        else
        {
            Exp = 1;
        }
    }
    newTerm.Coef = Coef;
    newTerm.Exp = Exp;
    addterm(ret,newTerm);
}
else
{
    substring = a.substr(prev,idx - prev + 1);
    transfer<<substring;
    transfer>>Coef;
    Exp = 0;
    newTerm.Coef = Coef;
    newTerm.Exp = Exp;
    addterm(ret,newTerm);
}
prev = StorePlus[i] + 1;
}
//处理最后一项。
int last = (int)StorePlus.size() - 1;
int last_index = StorePlus[last];
bool FindX = 0;
for (int i = last_index; i < a.size(); ++i)
{
    if(a[i] == 'x')
    {
        FindX = 1;
        idx = i;
        break;
    }
}
if(FindX)
{
    if(a[idx-1] == '+')
    {
        Coef = 1;
    }
    else
    {
        substring = a.substr(last_index + 1,idx - last_index - 1);
        stringstream ss;
```

```
            ss.clear();
            ss<<substring;
            ss>>Coef;
        }

        if(a[idx+1] == '^')
        {
            substring = a.substr(idx+2,a.size() - idx - 2);
            stringstream ss;
            ss.clear();
            ss<<substring;
            ss>>Exp;
        }
        else
        {
            Exp = 1;
        }
        newTerm.Coef = Coef;
        newTerm.Exp = Exp;
        addterm(ret,newTerm);
    }
    return ret;
}

void InputPoly(List<Term> &a)
{
    // int t_a;
    // cout<<"Please Input The Number Of Terms in: ";
    // cin>>t_a;
    // for (int i = 0; i < t_a; ++i)
    // {
    //     double coef;
    //     int exp;
    //     Term newTerm;
    //     cout<<i+1<<"."<<"Coef = :";
    //     cin>>coef;
    //     cout<<i+1<<"."<<"Exp = :";
    //     cin>>exp;
    //     newTerm.Coef = coef;
    //     newTerm.Exp = exp;
    //     addterm(a,newTerm);
    // }
    // cout<<"You Have Inserted A Polynomials: ";
    // Display(a);
    // cout<<"\n";
    cout<<"Please Input The Polynomials: ";
    string poly;
    cin>>poly;
    a = Extract(poly);
    cout<<"You Have Inserted A Polynomials : ";
    Display(a);
    cout<<"\n";
```

```
}

List<Term> DeepCopy(List<Term> &source)
{
    List<Term> ret;
    ret.clear();
    List<Term>::iterator itr;
    for (itr = source.begin();itr != source.end(); ++itr)
    {
        ret.push_back(Term((*itr).Coef,(*itr).Exp));
    }
    return ret;
}



int main()
{
    vector<Term> PolynomialInVector;
    List<Term> p1,p2,ans,temp;
    //t1 = 3x^5 + 2x^4 + 1x^2
    //t2 = x^2;
    Term newTerm(3,5);
    addterm(p1,newTerm);
    p1.push_back(Term(6,3));
    p1.push_back(Term(9,0));
    p2.push_back(Term(4,2));
    p2.push_back(Term(2,0));
    freopen("in","r",stdin);
    freopen("out","w",stdout);
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    while (1)
    {
        string choose;
        List<Term> a,b,ans;
        cout<<"Welcome to calculator of polynomials\n";
        cout<<"Enter Add To Get The Sum Of Two Polynomials\n";
        cout<<"Enter Sub To Subtract Two Polynomials\n";
        cout<<"Enter Mut To Mutiply Two Polynomials\n";
        cout<<"Enter Div To Divide A from B\n";
        cout<<"Enter Mod To Mod A from B\n";
        cout<<"Enter Get To Get Value In A Specified Polynomial\n";
        cout<<"Enter Other Button To Quit\n";
        cout<<"What's your choice : ";
        cin>>choose;
        cout<<"\n";
        if (StringToLower(choose) == "add")
        {
            a.clear();
            b.clear();
            InputPoly(a);
```

```
    InputPoly(b);
    ans = PolynomialsAdd(a,b);
    cout<<"The Ans of A + B = ";
    Display(ans);
    cout<<"\n";
}
else if (StringToLower(choose) == "sub")
{
    a.clear();
    b.clear();
    InputPoly(a);
    InputPoly(b);
    ans = PolynomialsSub(a,b);
    cout<<"The Ans of A - B = ";
    Display(ans);
    cout<<"\n";
}
else if (StringToLower(choose) == "mut")
{
    a.clear();
    b.clear();
    InputPoly(a);
    InputPoly(b);
    ans = PolynomialsMutipy(a,b);
    cout<<"The Ans of A * B = ";
    Display(ans);
    cout<<"\n";
}
else if (StringToLower(choose) == "div")
{
    a.clear();
    b.clear();
    InputPoly(a);
    InputPoly(b);
    List<Term> Keep_a,Keep_b;
    Keep_a = DeepCopy(a);
    Keep_b = DeepCopy(b);
    ans = try_divide(a,b);
    List<Term> remainder = try_mod(Keep_a,Keep_b);
    cout<<"The Ans of A / B = ";
    Display(ans);
    cout<<"The Remainder Polynomial = ";
    Display(remainder);
    cout<<"\n";
}
else if (StringToLower(choose) == "get")
{
    cout<<"Please Input The Polynoimal That You Want To Caculate\n";
    a.clear();
    InputPoly(a);
    cout<<"Please Input The X0 That You Want To Get: ";
    double x0;
```

```
            cin>>x0;
            cout<<"The Value Is: "<<CaculateValue(a,x0)<<"\n";
            cout<<"\n";
        }
        else
        {
            break;
        }
    }
    return 0;
}
```