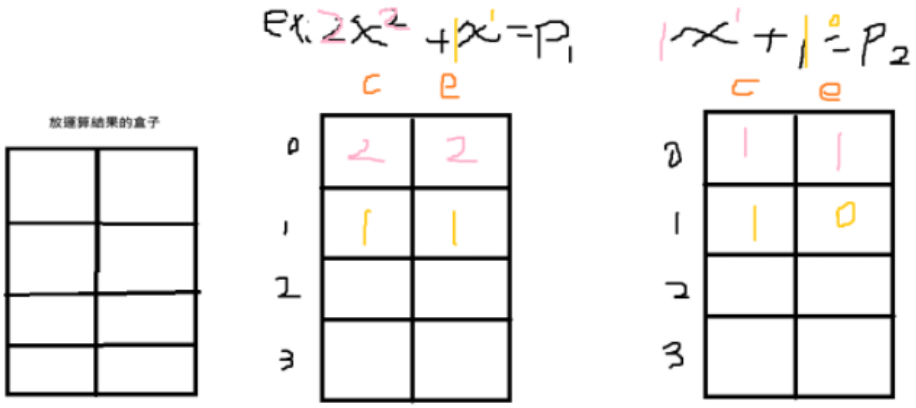
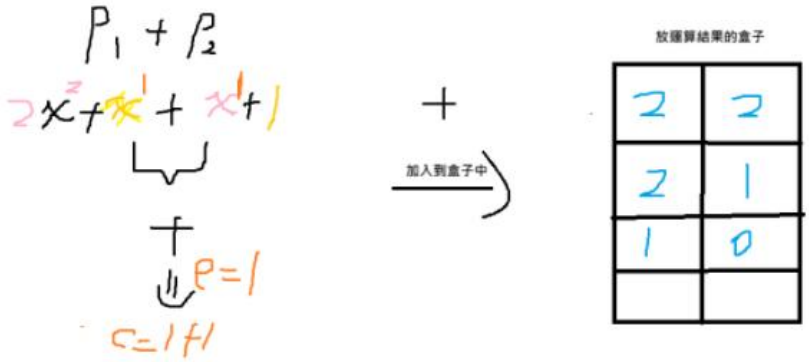
解題思路:

要先有三個盒子，有兩個來存 P1 和 P2 的資料，最後一個要來存放資料最後運算出的資料



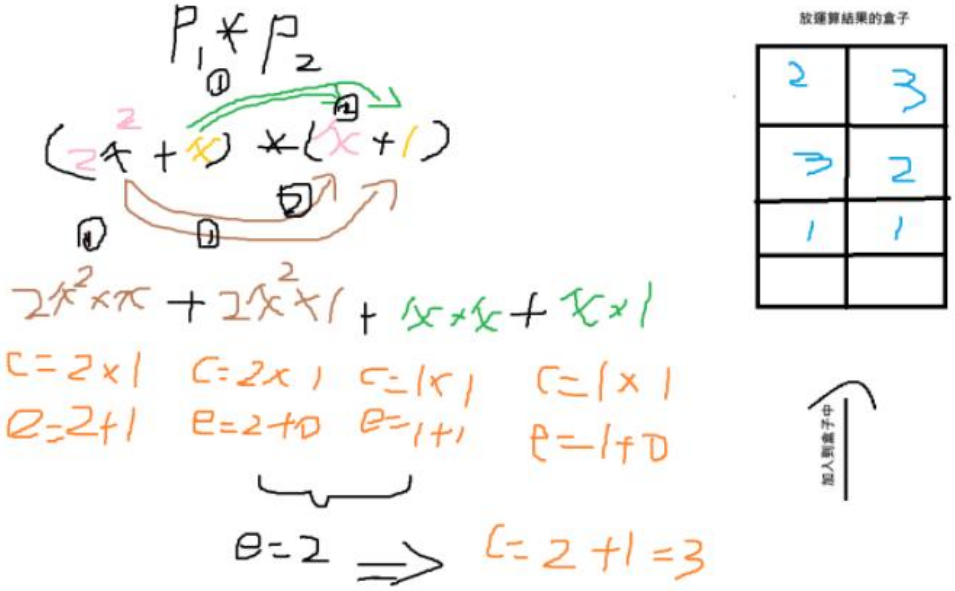
加法的思路:

如過 e 有一樣的值，就把 c 加來



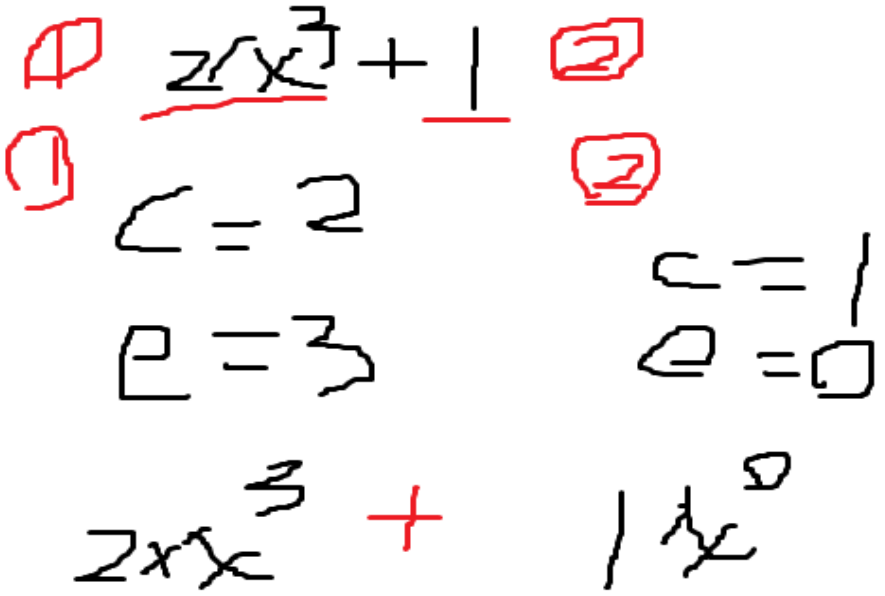
乘法的思路:

全部做一次相乘，如果有重複的 e 就把 c 加起來



求值:

先把 2X^3 拆開看=>2*(x^3)以此類推，在把多項式的每一項加起來



程式實作:

Class polynomial 的公開，我另外加了複製建構值，和擴建空間的函數

```
public:
    polynomial(int box = 10); // 新增位子
    ~polynomial(); // 歸還位子
    polynomial(polynomial &poly); //複製建構函數
    polynomial Add(polynomial poly); // 加法
    polynomial Mult(polynomial poly); // 乘法
    void newA(float coe,int exp);//存放資料和擴建box
    friend ostream& operator<<(ostream& output, polynomial& p);
    friend istream& operator>>(istream& input, polynomial& p);
    float Eval(float f);
};
```

加法運算:

先建立一個類別來存取算出的結果值，判斷指數有沒有一樣如果一樣就相加，沒有看誰的指數比較大如果比較大先放進箱子中，一樣大 sumCoef 是存兩個佳來的常數也就是 c 再放到箱子中，最下面 while()檢查有沒有遺漏的值

```
polynomial polynomial::Add(polynomial poly) {
        polynomial result;
    int i = 0, j = 0;
    while (i < terms && j < poly.terms) {
        if (termArray[i].exp == poly.termArray[j].exp) {//指數一樣
            float sumCoef = termArray[i].coe + poly.termArray[j].coe;
            if (sumCoef != 0) result.newA(sumCoef, termArray[i].exp);//相加傳入到newA()裡面存放
            i++;
            j++;
        } else if (termArray[i].exp > poly.termArray[j].exp) {//大於傳入到newA()裡面存放
            result.newA(termArray[i].coe, termArray[i].exp);
            i++;
        } else {//小於傳入到newA()裡面存放
            result.newA(poly.termArray[j].coe, poly.termArray[j].exp);
            j++;
        }
    }
    while (i < terms) {//剩下的值存到newA()裡面存放
        result.newA(termArray[i].coe, termArray[i].exp);
        i++;
    }
    while (j < poly.terms) {//剩下的值存到newA()裡面存放
        result.newA(poly.termArray[j].coe, poly.termArray[j].exp);
        j++;
    }
    return result;
}
```

乘法運算:

先建立一個類別來存取算出的結果值，先常數相乘，指數相加，檢查相乘後的結果中有沒有一樣的指數，for()是跑有沒有相同一樣的指數 if()跟新的指數跟舊的指數做相比完的值再存到箱子中，found 是檢查相加完會空下一個位子所以不要讓位子多一個出來。

```
polynomial polynomial::Mult(polynomial poly) {
    polynomial result;
    for (int i = 0; i < terms; i++) {
        for (int j = 0; j < poly.terms; j++) {
            double newCoef = termArray[i].coe * poly.termArray[j].coe;
            int newExp = termArray[i].exp + poly.termArray[j].exp;

            bool found = false; //檢查有沒有相同，如果有加並不要存到裡面
            for (int k = 0; k < result.terms; k++) {
                if (result.termArray[k].exp == newExp) {
                    result.termArray[k].coe += newCoef;
                    found = true;
                    break;
                }
            }
            if (!found) result.newA(newCoef, newExp);//沒有重複的值
        }
    }
    return result;
}
```

求值:

多項式值=c 相乘 X 次方 e

```
float polynomial::Eval(float x)  {
    double result = 0.0;
    for (int i = 0; i < terms; i++) {
        result += termArray[i].coe * pow(x, termArray[i].exp);//算輸多項式的值
    }
    return result;
}
```

複製建構函數:

創新的陣列，舊的陣列複製到新的陣列中在更新 capacity 和 terms，簡單來說傳參考要傳本尊進去函數中

```
polynomial::polynomial(polynomial &poly) {
    termArray = new Term[poly.capacity];
    for(int a=0;a<poly.terms;a++)termArray[a]=poly.termArray[a];
    capacity=poly.capacity;
    terms=poly.terms;
}
```

擴建空間:

```
void polynomial::newA(float coe, int exp) {//增加空間
    if(capacity==terms) {
        capacity*=2;//空間增加*2
        Term *newA=new Term[capacity];
        for(int i=0;i<terms;i++) {
            newA[i]=termArray[i];
        }
        delete []termArray; //把指標的值歸還
        termArray=newA;//指向新陣列
    }
    termArray[terms].coe=coe;//存入係數
    termArray[terms].exp=exp;//存入指數
    terms++;
}
```

Output:

For()看 terms 項目數量印出多少值，if()第一個值和最一個位子不會有加

```
ostream& operator<<(ostream& output, polynomial& poly) {
        for (int i = 0; i < poly.terms; i++) {
            if (i > 0 && poly.termArray[i].getcoe() > 0) output << "+";//後一個不會有+
            output << poly.termArray[i].getcoe() << "x^" << poly.termArray[i].getexp();
        }
    return output;
}
```

Input:

先出入項目數量，for()一個一個存到箱子中

```
istream& operator>>(istream& input, polynomial& poly) {
    int termCount;
    cout << "請輸入多項式的項數:";
    input >> termCount;
    for (int i = 0; i < termCount; i++) {//一個一個存
        double coef;
        int exp;
        cout << "請輸入係數與指數（如 2 3 表示 2x^3）:";
        input >> coef >> exp;
        poly.newA(coef, exp); //新增到陣列
    }
    return input;
}
```

完整程式

```
#include<iostream>
#include <cmath>//用在Eval()的計算
using namespace std;
class polynomial ;
class Term {
    friend polynomial;
    float coe;
    int exp;
public:// 不可以直接使用coe和exp
    float getcoe(){return coe;} //用函式來給予
    int getexp(){return exp;}
};
class polynomial {
private:
    Term *termArray;
    int capacity; // 容量
    int terms; // 項目數量
public:
    polynomial(int box = 10); // 新增位子
    ~polynomial(); // 歸還位子
    polynomial(polynomial &poly); //複製建構函數
    polynomial Add(polynomial poly); // 加法
    polynomial Mult(polynomial poly); // 乘法
    void newA(float coe,int exp);//存放資料和擴建box
    friend ostream& operator<<(ostream& output, polynomial& p);
    friend istream& operator>>(istream& input, polynomial& p);
    float Eval(float f);
};
polynomial::polynomial(int box):capacity(box),terms(0) {
    termArray=new Term[box];
}
```

```
28      };
29      polynomial::polynomial(int box):capacity(box),terms(0) {
30          termArray=new Term[box];
31      }
32      polynomial::~polynomial() {
33          delete []termArray;//解構
34      }
35      void polynomial::newA(float coe, int exp){...}
49      polynomial::polynomial(polynomial &poly){...}
55
56      polynomial polynomial::Add(polynomial poly){...}
83
84      polynomial polynomial::Mult(polynomial poly){...}
104
105     float polynomial::Eval(float x){...}
112
113     ostream& operator<<(ostream& output, polynomial& poly){...}
120
121
122     istream& operator>>(istream& input, polynomial& poly){...}
135
136     int main() {
137         polynomial p1, p2;
138         cout << "第一個多項式" << endl;
139         cin >> p1;
140         cout << "第二個多項式" << endl;
141         cin >> p2;
142         cout << "第一個多項式：" << p1 << endl;
143         cout << "第二個多項式：" << p2 << endl;
144         polynomial sum = p1.Add(p2);
145         cout << "兩多項式相加結果：" << sum << endl;
```

```
146         polynomial product = p1.Mult(p2);
147         cout << "兩多項式相乘結果：" << product << endl;
148         double x;
149         cout << "請輸入 x 的值以計算多項式值：";
150         cin >> x;
151         cout << "第一個多項式在 x = " << x << " 的值為：" << p1.Eval(x) << endl;
152         cout << "第二個多項式在 x = " << x << " 的值為：" << p2.Eval(x) << endl;
153         return 0;
154     }
155
```

效能分析:

Time complexity:

加法:O(m+n) ，m 是第一個多項式(P1)、n 是第二多項式(P2)

乘法:O(m*n*k) m 是第一個多項式(P1)、n 是第二多項式(P2)、k 是結果多項式的項數(c)

Eval: O(1*m) m 是多項式的項數(c)、1 是 pow()

Space complexity:

動態陣列儲存的空間為 O(t)， t 是多項式的最大項數。

加法與乘法操作需要額外的暫存空間 O(m*n)。m=P1、n=P2

效能量測:

P1=3x^2+2x+1　P2=−x^2+4x

加法結果:2X^2+6X+1

乘法結果: -3x^4+10x^3+7x^2+4x^1

X=3

求值結果:P1=34，P2=3

```
第一個多項式
請輸入多項式的項數:3
請輸入係數與指數（如 2 3 表示 2x^3）:3 2
請輸入係數與指數（如 2 3 表示 2x^3）:2 1
請輸入係數與指數（如 2 3 表示 2x^3）:1 0
第二個多項式
請輸入多項式的項數:2
請輸入係數與指數（如 2 3 表示 2x^3）:-1 2
請輸入係數與指數（如 2 3 表示 2x^3）:4 1
第一個多項式：3x^2+2x^1+1x^0
第二個多項式：-1x^2+4x^1
兩多項式相加結果：2x^2+6x^1+1x^0
兩多項式相乘結果：-3x^4+10x^3+7x^2+4x^1
請輸入 x 的值以計算多項式值：3
第一個多項式在 x = 3 的值為：34
第二個多項式在 x = 3 的值為：3
```

申論及開發報告:

程式的時間複雜度在加法操作中是 O(n+m)，乘法操作則是 O(n*m*k)，其中 n 和 m 分別是兩個多項式的項數。這樣的複雜度在小型多項式運算中是可接受的，但在項數巨大的情況下可能需要進一步優化，結合了基礎資料結構和算法的應用，為多項式運算的進一步研究扎實的基礎。