國立台灣海洋大學資訊工程系 C++ 程式設計 期中考參考答案

姓名：_____學號：_____ 102/04/30

考試時間：**10:00 – 12:00**

試題敘述蠻多的，看清楚題目問什麼，針對重點回答，總分有 **125**，請看清楚每一題所佔的分數再回答

考試規則：1. 不可以翻閱參考書、作業及程式
2. 請勿使用任何形式的電腦 (包含手機、計算機、相機以及其它可運算或是連線的電子器材)
3. 請勿左顧右盼、請勿交談、請勿交換任何資料、試卷題目有任何疑問請舉手發問 (看不懂題目不見得是你的問題，有可能是中英文名詞的問題)、最重要的是隔壁的答案可能比你的還差，白卷通常比錯得和隔壁一模一樣要好
4. 提早繳卷同學請直接離開教室，請勿逗留喧嘩
5. 違反上述任何一點之同學一律送請校方處理
6. 繳卷時請繳交***簽名過之試題卷及答案卷***

在一個應用軟體中，需要使用許多高次數的一元多項式來進行運算，例如 $f(x)=1.5x^{25}-2.3x^{11}+x^{7}$，因此設計一個 HiDPoly 類別，每一個 HiDPoly 類別的物件代表一個 n 次實數係數的一元多項式，這樣的物件需要有一些基本的資料設定、運算、與驗證功能，如下圖程式所示：

```
1. ifstream infile("poly.dat");      // 開啓檔案串流
2. HiDPoly poly1(infile),            // 由檔案串流中初始化
3.    poly2(infile), poly3(infile); // poly1, poly2, 及 poly3
4.
5. poly1.print("poly1 = ", cout);
6. poly2.print("poly2 = ", cout);
7.
8. HiDPoly poly4 = poly1;       // 由 poly1 初始化 poly4
```

```
 9. poly4.addEqual(poly2);          // poly4 = poly4 + poly2
10. poly4.print("poly1+poly2=", cout);   //列印和多項式
11. assert(poly4.equals(poly3, 1e-12)); // 驗證加法正確性
12.
13. poly4 = poly1;                  // 將 poly4 重設爲 poly1
14. poly4.mulEqual(poly2);          // poly4 = poly4 * poly2;
15. poly4.print("poly1*poly2=", cout);   // 列印乘積多項式
16. assert(poly4.equals(poly1, poly2));   // 驗證乘法正確性
```

檔案 poly.dat 內容如下：

```
4
8 2.1
5 3.5
2 5.4
0 0.6
5
12 4.1
10 6.5
4 2.4
2 1.6
1 -3.5
8
12 4.1
10 6.5
8 2.1
5 3.5
4 2.4
2 7
1 -3.5
0 0.6
```

程式輸出結果如下：

```
poly1 = 2.1*x^8 + 3.5*x^5 + 5.4*x^2 + 0.6*x^0
poly2 = 4.1*x^12 + 6.5*x^10 + 2.4*x^4 + 1.6*x^2 + -3.5*x^1
poly1+poly2 =
4.1*x^12 + 6.5*x^10 + 2.1*x^8 + 3.5*x^5 + 2.4*x^4 + 7*x^2 + -3.5*x^1 + 0.6*x^0
poly1*poly2 =
8.61*x^20 + 13.65*x^18 + 14.35*x^17 + 22.75*x^15 + 22.14*x^14 + 42.6*x^12 +
7.26*x^10 + 1.05*x^9 + 5.6*x^7 + 0.71*x^6 + 10.08*x^4 + -18.9*x^3 + 0.96*x^2 +
-2.1*x^1
```

請回答下列問題，依序完成 HiDPoly 類別(HiDPoly.h 及 HiDPoly.cpp)的設計

1. [5] 根據上面這一段程式的內容，包含這一段程式的檔案需要引入哪些標頭檔?

Sol:

HiDPoly.h, fstream, assert.h, iostream,

2. [10] 根據上面這一段程式的內容，這個 HiDPoly 類別至少應該有一個建構元 (constructor) 函式由輸入串流讀入多項式，一個 addEqual() 成員函式負責將兩個多項式相加起來，一個 mulEqual() 成員函式負責將兩個多項式相乘起來，一個 print() 成員函式負責列印一個字串訊息以及多項式，兩個 equals() 成員函式，請在 HiDPoly.h 檔案中宣告這個類別，以及這幾個成員函式的原型 (prototype)，請根據上述程式設計各個參數的型態、回傳值的型態、以及存取權限 (public or private)，可以使用 const 的地方請盡量使用

Sol:

```
#include <iostream>
#include <fstream>

class HiDPoly
```

```
{
public:
    HiDPoly(std::ifstream &);
    void addEqual(const HiDPoly &);
    void mulEqual(const HiDPoly &);
    void print(char msg[], std::ostream &) const;
    bool equals(const HiDPoly &rhs, const double perror) const;
    bool equals(const HiDPoly &poly1, const HiDPoly &poly2) const;
};
```

3. [5] 接上題，因為題目希望這個類別的物件可以存放高次數的一元多項式，也沒有限制最高次數是多少，看上面的例子又發現可能是很稀疏的高次多項式，例如 $f(x)=1.5x^{45}-2.3x^{11}$，所以我們希望在 HiDPoly 類別**內**定義一個有兩個欄位 degree 和 coef 的結構 DegreeCoef 來描述多項式的每一項：例如 (45,1.5) 和(11,-2.3)，請定義此內部結構(inner struct/class)

Sol:

```
class HiDPoly
{
public:
    struct DegreeCoef
    {
        int degree;
        double coef;
    };
};
```

4. [5] HiDPoly 類別的物件需要有兩個資料成員，其一用來存放多項式有幾項，另一個成員是運用標準函式庫中的 vector 來存放**動態配置**的 DegreeCoef 結構的指標，請在類別中定義此兩個成員並說明 HiDPoly.h 檔案中需要的引入檔

Sol:

```
…
#include <vector>

class HiDPoly
{
public:
    struct DegreeCoef
    {
        int degree;
        double coef;
    };
    …
private:
    int m_nterms;
    std::vector<DegreeCoef *> m_terms;
};
```

5. [5] 請撰寫預設建構元 (default constructor) 成員函式，運用初始化串列 (initialization list) 將多項式項數設為 0

Sol:

```
------------ HiDPoly.h ------------
…
class HiDPoly
{
public:
    …
    HiDPoly();
    …
};
------------ HiDPoly.cpp ------------
HiDPoly::HiDPoly():m_nterms(0)
{
}
```

6. [10] 請撰寫由資料檔案串流中讀取多項式的建構元函式，資料的格式如上圖 poly.dat 所

示，首先是一個整數代表多項式的項數，接下來每一列資料代表多項式的一項 (次數, 係數) (例如 8 2.1 代表 2.1 x^8 這一項)，在檔案中每一個多項式都是先列出高次項然後才列出低次項，在物件中的 vector 成員內也需要依照這種順序排放各項的次數及係數

Sol:

```
---------- HiDPoly.h ----------
#include <fstream>
#include <vector>

class HiDPoly
{
public:
        …
        HiDPoly(std::ifstream &);
        …
private:
        int m_nterms;
        std::vector<DegreeCoef *> m_terms;
};
---------- HiDPoly.cpp ----------
#include "HiDPoly.h"
#include <fstream>
#include <vector>
using namespace std;

HiDPoly::HiDPoly(ifstream &ins)
{
        int i;
        DegreeCoef *term;

        ins >> m_nterms;
        for (i=0; i<m_nterms; i++)
        {
                term = new DegreeCoef;
                ins >> term->degree >> term->coef;
                m_terms.push_back(term);
        }
}
```

7. [20] 請撰寫多項式相加的成員函式 addEqual(const HiDPoly &rhs)，相加的結果修改原來物件中的多項式，例如(2.1 x^8 + 3.5 x^5 + 5.4 x^2 + 0.6) + (4.1 x^{12} + 6.5 x^{10} + 2.4 x^4 + 1.6 x^2 -3.5 x) = 4.1 x^{12} + 6.5 x^{10} + 2.1 x^8 + 3.5 x^5 + 2.4 x^4 + 7 x^2 -3.5 x + 0.6，請注意相加以後的多項式的內部資料還是需要由高次項排到低次項，相同次數的係數需要加起來，演算法的效率不需要特別考慮，以正確完成功能為優先目標

Sol:

```
---------- HiDPoly.h ----------
…
#include <vector>

class HiDPoly
{
public:
        …
        void addEqual(const HiDPoly &);
        …
private:
        int m_nterms;
        std::vector<DegreeCoef *> m_terms;
};
---------- HiDPoly.cpp ----------
#include "HiDPoly.h"
…
#include <vector>
using namespace std;
```

```
void HiDPoly::addEqual(const HiDPoly &rhs)
{
    int i=0, j=0;
    HiDPoly sum;

    while ((i<m_nterms)&&(j<rhs.m_nterms))
    {
        if (m_terms[i]->degree > rhs.m_terms[j]->degree)
        {
            sum.m_terms.push_back(new DegreeCoef(*(m_terms[i++])));
            sum.m_nterms++;
        }
        else if (m_terms[i]->degree < rhs.m_terms[j]->degree)
        {
            sum.m_terms.push_back(new DegreeCoef(*(rhs.m_terms[j++])));
            sum.m_nterms++;
        }
        else
        {
            sum.m_terms.push_back(new DegreeCoef(*(m_terms[i++])));
            sum.m_terms[sum.m_nterms++]->coef += rhs.m_terms[j++]->coef;
        }
    }
    if (i==m_nterms)
        while (j<rhs.m_nterms)
        {
            sum.m_terms.push_back(new DegreeCoef(*(rhs.m_terms[j++])));
            sum.m_nterms++;
        }
    else
        while (i<m_nterms)
        {
            sum.m_terms.push_back(new DegreeCoef(*(m_terms[i++])));
            sum.m_nterms++;
        }

    *this = sum;   // 此敘述需要第 11 題的 assignment operator
}
```

8. [15] 不論你前一題 addEqual() 成員函式有沒有寫出來，請運用前一題的 addEqual 成員函式撰寫多項式相乘的成員函式 mulEqual(const HiDPoly &rhs)，相乘的結果修改原來物件中的多項式，例如$(2.1\ x^8 + 3.5\ x^5 + 5.4\ x^2 + 0.6)\ (4.1\ x^{12} + 6.5\ x^{10} + 2.4\ x^4 + 1.6\ x^2 - 3.5\ x) = 8.61\ x^{20} + 13.65\ x^{18} + 14.35\ x^{17} + 22.75\ x^{15} + 22.14\ x^{14} + 42.6\ x^{12} + 7.26\ x^{10} + 1.05\ x^9 + 5.6\ x^7 + 0.71\ x^6 + 10.08\ x^4 - 18.9\ x^3 + 0.96\ x^2 - 2.1\ x$，請注意相乘以後的多項式的內部資料還是需要由高次項排到低次項，相同次數的係數需要加起來

Sol:

```
------------- HiDPoly.h -------------
…
#include <vector>

class HiDPoly
{
public:
    …
    void mulEqual(const HiDPoly &);
    …
private:
    int m_nterms;
    std::vector<DegreeCoef *> m_terms;
};
------------- HiDPoly.cpp -------------
#include "HiDPoly.h"
…

void HiDPoly::mulEqual(const HiDPoly &rhs)
{
```

```
    HiDPoly sum;
    int i, j;
    for (j=0; j<rhs.m_nterms; j++)
    {
        HiDPoly tmp(*this);
        for (i=0; i<tmp.m_nterms; i++)
        {
            tmp.m_terms[i]->degree += rhs.m_terms[j]->degree;
            tmp.m_terms[i]->coef *= rhs.m_terms[j]->coef;
        }
        sum.addEqual(tmp);
    }
    *this = sum; // 此敘述需要第 11 題的 assignment operator
}
```

9. [5] 請實作一個 print 成員函式列印出上圖的結果

Sol:

```
------------- HiDPoly.h -------------
#include <iostream>
…

class HiDPoly
{
public:
    struct DegreeCoef
    {
        int degree;
        double coef;
    };
    …
    void print(char msg[], std::ostream &) const;
    …
private:
    int m_nterms;
    std::vector<DegreeCoef *> m_terms;
};
------------- HiDPoly.cpp -------------
#include "HiDPoly.h"
#include <iostream>
…
using namespace std;

void HiDPoly::print(char msg[], ostream &os) const
{
    int i;
    os << msg;
    for (i=0; i<m_nterms-1; i++)
        os << m_terms[i]->coef << "*" << "x^" << m_terms[i]->degree << " + ";
    os << m_terms[i]->coef << "*" << "x^" << m_terms[i]->degree << endl;
}
```

10. [10] 由於 HiDPoly 類別的資料成員包含動態配置的資料，所以請撰寫一個拷貝建構元成員函式，另外請問不撰寫拷貝建構元常常會遇見的錯誤是什麼?

Sol:

```
------------- HiDPoly.h -------------
…
#include <vector>

class HiDPoly
{
public:
    …
    HiDPoly(HiDPoly &src);
    …
private:
    int m_nterms;
    std::vector<DegreeCoef *> m_terms;
};
```

```
---------- HiDPoly.cpp ----------
#include "HiDPoly.h"
…
#include <vector>
using namespace std;

HiDPoly::HiDPoly(HiDPoly &src):m_nterms(src.m_nterms)
{
      int i;
      for (i=0; i<m_nterms; i++)
            m_terms.push_back(new DegreeCoef(*(src.m_terms[i])));
}
```

==========

通常會遇見的錯誤是 Dangling reference, 就是當兩個物件都指向相同的自行配置記憶體時, 如果有一個先釋放掉, 就會使得另外一個物件裡面的指標事實上是指著已經被釋放掉的記憶體

11. [10] 與上題相同的原因，請製作一個設定運算子 operator=() 成員函式以支援上圖中第 13 列複製一個 HiDPoly 物件(你可以使用 vector<type>::erase(vector<type>::iterator start, vector<type>::iterator end) 成員函式來刪除 vector<type> 物件中所有的資料)

Sol:

```
---------- HiDPoly.h ----------
…
#include <vector>

class HiDPoly
{
public:
      …
      HiDPoly& operator=(const HiDPoly &rhs);
      …
private:
      int m_nterms;
      std::vector<DegreeCoef *> m_terms;
};
---------- HiDPoly.cpp ----------
#include "HiDPoly.h"
…
#include <vector>
using namespace std;

HiDPoly& HiDPoly::operator=(const HiDPoly &rhs)
{
      int i;
      if (this == &rhs)
            return *this;

      for (i=0; i<m_nterms; i++)
            delete m_terms[i];
      m_terms.erase(m_terms.begin(), m_terms.end());

      for (i=0; i<rhs.m_nterms; i++)
            m_terms.push_back(new DegreeCoef(*(rhs.m_terms[i])));
      m_nterms = rhs.m_nterms;

      return *this;
}
```

12. [5] 與上題相同的原因，請撰寫解構元(destructor)函式以避免 memory leakage

Sol:

```
---------- HiDPoly.h ----------
…
#include <vector>

class HiDPoly
{
```

```
public:
        …
        virtual ~HiDPoly();
        …
private:
        int m_nterms;
        std::vector<DegreeCoef *> m_terms;
};
------------- HiDPoly.cpp -------------
#include "HiDPoly.h"
…

HiDPoly::~HiDPoly()
{
        int i;
        for (i=0; i<m_terms.size(); i++)
                delete m_terms[i];
}
```

13. [10] 請撰寫驗證加法正確性的 equals() 成員函式，這個函式的參數需要包括一個 HiDPoly 物件的參考以及一個誤差的範圍，這個函式裡請你比較兩個多項式裡每一個係數是否都在指定的誤差範圍內 (請注意兩個多項式因為經過計算，有可能項數並不相同，有可能有一個多項式有 10^{-13} x^5 這種係數很小的資料項，而另一個多項式根本沒有 x^5 項，就算出現這種狀況，只要誤差在指定的範圍內，就算是相同的)

Sol:

```
------------- HiDPoly.h -------------
…
#include <vector>

class HiDPoly
{
public:
        …
        bool equals(const HiDPoly &rhs, const double perror) const;
        …
private:
        int m_nterms;
        std::vector<DegreeCoef *> m_terms;
};
------------- HiDPoly.cpp -------------
#include "HiDPoly.h"
…
#include <math.h>
…

bool HiDPoly::equals(const HiDPoly &rhs, const double perror) const
{
        int i=0, j=0;

        while ((i<m_nterms)&&(j<rhs.m_nterms))
        {
                if (m_terms[i]->degree > rhs.m_terms[j]->degree)
                {
                        if (fabs(m_terms[i++]->coef) > perror)
                                return false;
                }
                else if (m_terms[i]->degree < rhs.m_terms[j]->degree)
                {
                        if (fabs(rhs.m_terms[j++]->coef) > perror)
                                return false;
                }
                else
                {
                        if (fabs(m_terms[i++]->coef - rhs.m_terms[j++]->coef) > perror)
                                return false;
                }
        }
```

```
    if (i==m_nterms)
        while (j<rhs.m_nterms)
        {
            if (fabs(rhs.m_terms[j++]->coef) > perror)
                return false;
        }
    else /*if (j==rhs.m_nterms)*/
        while (i<m_nterms)
        {
            if (fabs(m_terms[i++]->coef) > perror)
                return false;
        }
    return true;
}
```

14. [5] 請撰寫一個 evaluate() 成員函式，參數是一個 double 型態的 x 數值，此成員函式可以運用標準 C/C++ 的 math.h 函式庫中的 double pow(double base, double exponent) 函式來計算 f(x) 的數值

Sol:

```
---------- HiDPoly.h ----------
…
#include <vector>

class HiDPoly
{
public:
    …
    double evaluate(double) const;
    …
private:
    int m_nterms;
    std::vector<DegreeCoef *> m_terms;
};
---------- HiDPoly.cpp ----------
#include "HiDPoly.h"
…

double HiDPoly::evaluate(const double x) const
{
    int i;
    double sum=0.0;
    for (i=0; i<m_nterms; i++)
        sum += m_terms[i]->coef * pow(x, m_terms[i]->degree);
    return sum;
}
```

15. [5] 請撰寫驗證乘法正確性的 equals() 成員函式驗證 f(x)=g(x)h(x)，這個函式的參數需要包括兩個 HiDPoly 物件 g(x)與 h(x) 的參考，這個成員函式裡請你運用標準 C/C++的 stdlib.h 函式庫中的 rand()函式隨機挑選 1000 個**浮點數** x，分別計算 f(x)、g(x)、與 h(x), 並且驗證浮點數 g(x)*h(x)和 f(x)的差值是否在 f(x)***10**$^{-10}$ 的相對誤差範圍內，如果 1000 個中發現任何一個錯誤就算是驗證失敗，回傳 false

Sol:

```
---------- HiDPoly.h ----------
…
#include <vector>

class HiDPoly
{
public:
    …
    bool equals(const HiDPoly &poly1, const HiDPoly &poly2) const;
    double evaluate(double) const;
    …
private:
    int m_nterms;
```

```
        std::vector<DegreeCoef *> m_terms;
};
------------- HiDPoly.cpp -------------
#include "HiDPoly.h"
…
#include <stdlib.h>
#include <math.h>
…

bool HiDPoly::equals(const HiDPoly &poly1, const HiDPoly &poly2) const
{
        int i;
        double x, y1, y2, y3;
        for (i=0; i<1000; i++)
        {
                x = rand() * 2.5 / RAND_MAX;
                y1 = evaluate(x);
                y2 = poly1.evaluate(x);
                y3 = poly2.evaluate(x);
                if (fabs(y1-y2*y3) > fabs(y1)*1e-10)
                        return false;
        }
        return true;
}
```