UEE1303 S18: Midterm Examination
-- Programming Part --

FULL SCORES: 120 points

ID NUMBER（學號）: _____

FULL NAME（姓名）: _____

Problem Set 1 (20 points)		**Problem Set 2** (30 points)			**Problem Set 3** (20 points)	
題號	助教簽名	題號	助教簽名		題號	助教簽名
01-1		02-1			03	
01-2						
Problem Set 4 (80 points)		02-2			**Problem Set 5** (30 points)	
04-1					05	
04-2					**Total Score**	
04-3						
04-4						

UEE1303 S18: Midterm Examination

-- Programming Part --

FULL SCORES:
120%

EXAMINATION TIME:
180 minutes

INSTRUCTIONS:

This midterm exam is composed of 5 different problem sets. You are allowed to open any notes or books, but prohibited to browse on the internet to search C/C++ codes as direct answers. Read carefully the statements and requirements of each problem. Once you complete your program for one problem, please raise your hand and TA will come to you and test your program. Please note that points will be given until your program fully fulfills the requirements of each problem. No partial credits will be given.

You should be aware of that some lengthy problems are not as difficult as you think. In contrast, some short problems may not be solved easily. Last but not least, **cheating** is a serious academic demeanor and should be avoided at all most. Once any cheating is caught, all your answers will be void and get 0 point for your midterm. **Good luck!**

【PROBLEM SET 1】(20%)

Design a class **Box** that has double variables for the lengths of the three sides of a box in private data.

1. You should get the information of box from the file. (10%)
2. You should implement three functions that includes printVolume, printMinSide and printMaxSide. (10%)
   - printVolume() prints the volume of the box.
   - printMinSide() prints the smallest side length.
   - printMaxSide() prints the largest side length.

Note that you cannot modify the main.cpp.

The main function is as the following:

```
int main(int argc, char *argv[])
{
    ifstream infile(argv[1]);
    Box box = ReadFile(infile);
    box.printVolume();
    box.printMinSide();
    box.printMaxSide();
    return 0;
}
```

The output should be

Case1

```
>./pg01 case1
The Volume is 72000.00
The Min Side is 20.00
The Max Side is 90.00
```

Case2

```
>./pg01 case2
The Volume is 85473.75
The Min Side is 12.50
The Max Side is 198.20
```

Case3

```
>./pg01 case3
The Volume is 41063.62
The Min Side is 34.50
The Max Side is 34.50
```

【PROBLEM SET 2】(30%)

Please download p2.cpp first. The program contains two classes. In this problem, you need to finish the program with following requirements:

Class Date: (10%)

- Stores a date in to year, month (1-12) and day (1-31)
- The default constructor sets a date as 1900/1/1. (1%)
- The function setDate() sets a date in the form YYYYMMDD. (5%)
- The function **getYear()** returns the year. (1%)
- The function **print()** displays the date in the format of YYYY/MM/DD (3%)

Class Profile: (20%)

- Stores a personal information includes first name, last name, gender, birthday, age and id.
- The default constructor sets a profile with name "John Doe" and gender "M". (2%)
- Another two overloaded constructors need to be completed. The arguments are first name/last name/gender and first name/last name/gender/birthday respectively. The format of birthday is YYYYMMDD. (6%)
- The function setAge() computes the person's age by the birthday year and the current year. For example, if a man was born in 1993 and the current year is 2018, his age is 2018 – 1993 = 25. **You need to get the current year from the system.** (5%)
- The function setID() generates id randomly. For a male, the id is between 1000000 and 1999999. For a female, the id is between 2000000 and 2999999. (5%)
- The function print() displays the profile as shown in Sample Output. (2%)

The output should be

```
Name: John Doe
Gender: M
Birthday: 1900/1/1
Age: 118
ID: 1375458
Name: Charles Wen
Gender: M
Birthday: 1900/1/1
```

```
Age: 118
ID: 1942445
Name: Shohei Ohtani
Gender: M
Birthday: 1994/7/5
Age: 24
ID: 1421177
Name: Aragaki Yui
Gender: F
Birthday: 1988/6/11
Age: 30
ID: 2937474
Name: Ashida Mana
Gender: F
Birthday: 2004/6/23
Age: 14
ID: 2235717
```

Note that you cannot modify the main.cpp.

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstdlib>
#include <ctime>

using namespace std;
class Date
{
public:
    Date();
    void print();
    void setDate(string);
    int getYear();
```

```
private:
    int month;
    int day;
    int year;
};

class Profile
{
public:
    Profile();
    Profile(string, string, char);
    Profile(string, string, char, string);
    void setAge();
    void setID();
    void print();
private:
    string    firstName;
    string    lastName;
    char      gender;
    Date      birthday;
    int       age;
    int       id;
};

int main()
{

    srand(time(NULL));

    Profile person1;
    person1.print();

    Profile person2("Charles", "Wen", 'M');
    person2.print();

    Profile person3("Shohei", "Ohtani", 'M', "19940705");
```

```
    person3.print();

    Profile person4("Aragaki", "Yui", 'F', "19880611");
    person4.print();

    Profile person5("Ashida", "Mana", 'F', "20040623");
    person5.print();

    return 0;
}
```

【PROBLEM SET 3】(20%)

The **Euclidean distance** between two points **p** and **q** is defined as:

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}.$$

where *n* indicates a n-dimensional space. In this problem, a **Line** object consists of two **Point** object, and the distance between these two points can be calculated by calling corresponding function.

In the following, the distance between two points in 2-dimensional and 3-dimensional spaces are calculated. Please implement the member function (constructor, setPoint(), calDist(), and printDist()) of the class **Line** and **Point**.

Note that pg03.cpp and pg03.h are provided and you cannot modify them.

The main function is as the following:

```
int main()
{
    // two dimensional points
    Line a(2);

    // cordinates of two points
    double point11[2] = {0,0}, point12[2] = {3,4};

    // set the cordinates
    a.setPoint(point11, point12);

    // calculate the distance and print it out
    a.calDist();
    a.printDist();

    // three dimensional points
    Line b(3);
```

```
    // cordinates of two points
    double point21[3] = {0,0,0}, point22[3] = {3,4,5};

    // set the cordinates
    b.setPoint(point21, point22);

    // calculate the distance and print it out
    b.calDist();
    b.printDist();

    return 0;
}
```

The output should be

```
The distance between points is: 5
The distance between points is: 7.07107
```

【PROBLEM SET 4】(80%)

Given the definitions of class **CVec** in pg04.h and the main program in main.cpp, please complete the tasks in each problem. You should complete the file in pg04.cpp and add proper prototypes/annotations on pg04.h.

```
//pg04.h
#ifndef __PG01_H__
#define __PG01_H__
#include <iostream>
using namespace std;
enum TLGC {L0, L1, LX};
class CVec {
private:
   TLGC *vec;                    //array in 3-valued logic
   int size;                  //size of the array
public:
   //Q1: Basics
   CVec();                       //size=0
   CVec(int);                    //randomize a user-specified TLGC vec
   CVec(int, TLGC*);             //given a size and an TLGC array
   CVec(const CVec&);            //copy constructor
   ~CVec();
   int getSize()        ;
   TLGC* getVec()       ;
   void Print()         ;
   //Q2: trancount and overload operator (!)

   //Q3: overload operators ( & and |)

   //Q4: overload operators (+ and <<)
};
#endif
```

PG04-1 BASICS (20 points)

Please complete all constructors and the required member functions in Q1.

```
TLGC m[4] = {LX,L1,L0,LX};
CVec a(2), b(5);
const CVec c(4,m);
CVec *px(&b), *py;
py = new CVec(c);
  ...
//Q1
cout << "[Q1]" << endl;
cout << "    ( a) " << a.getSize() << " " << a.getVec() << " "; a.Print(); cout << endl;
cout << "    ( b) " << b.getSize() << " " << b.getVec() << " "; b.Print(); cout << endl;
cout << "    ( c) " << c.getSize() << " " << c.getVec() << " "; c.Print(); cout << endl;
cout << "    (px) " << px->getSize() << " " << px->getVec() << " "; px->Print(); cout << endl;
cout << "    (py) " << py->getSize() << " " << py->getVec() << " "; py->Print(); cout << endl;
```

(1) There are four ways to construct a `CVec` object. One uses the default constructor `CVec()` to set size/vec as 0/NULL. One uses `CVec(int)` to randomize vec with a user-specified number of elements. Another uses `CVector(int,TLGC*)` to receive a TLGC array with its size. The other uses a copy constructor with `CVector(const CVector &)`.

(2) Destructor `~CVec()` needs to set size/vec as 0/NULL and cleans the space if allocated before.

(3) `getSize()` and `getVec()` are used to retrieve size and vec of the calling object where `Print()` is a normal member function to show the content of vec at the end on screen.

```
>./pg04 1
[Q1]
   ( a) 2 0x17e2c20 11
   ( b) 5 0x17e2c40 01X11
   ( c) 4 0x17e2c60 X10X
   (px) 5 0x17e2c40 01X11
   (py) 4 0x17e2ca0 X10X
>
```

PG04-2 (20 points)

Give Q2 in the main function for this problem,

```
//Q2
cout << "[Q2]" << endl;
const CVec d(15);
cout << "    ( d) "; d.Print();
cout << " w/ tran#=" << d.trancount() << endl;
cout << "    (!d) "; (!d).Print();
cout << " w/ tran#=" << (!d).trancount() << endl;
```

Please complete one member function and one overloading operator (!) to calculate the number of transitions in one CVec and the inversion of such object, then showing the corresponding information on screen using a member function Print().

(1) After removing x, any 0 to 1 or 1 to 0 in vec should be counted as a transition. Therefore, for d with vec as 001X1X1X10011XX, the total number of transitions is 3.

(2) Unary operator ! is overloaded to support the bitwise inversion of vec in the CVec object. For example, !(01X0)=10X1.

Therefore, !d = 110X0X0X01100XX.

(3) Note that you need to modify the statement properly to show the expected results.

```
>./pg04 2
[Q2]
   ( d) 001X1X1X10011XX w/ tran#=3
   (!d) 110X0X0X01100XX w/ tran#=3
>
```

PG04-3 (20 points)

Give Q3 in the main function for this problem,

```
//Q3
cout << "[Q3]" << endl;
CVec    p(5), q(4);
const CVec& e = b & p;
cout << "    ( e) "; b.Print(); cout << "&"; p.Print(); cout << "="; e.Print(); cout << endl;
CVec * pf = new CVec(c | q);
cout << "    (pf) "; c.Print(); cout << "|"; q.Print(); cout << "="; pf->Print(); cout << endl;
```

Please complete two overloading operators (`&` and `|`) to perform the bitwise **`AND`** and bitwise **`OR`** operations on two `CVec` objects, then showing the corresponding information on screen using `Print()`.

For example, given `a=01X1`, `b=1XX0`, `a&b=0XX0`, `a|b=11X1`

The expected result is shown as

```
> ./pg04 3
[Q3]
   ( e) 01X11&001X1=00XX1
   (pf) X10X|X1X1=X1X1
>
```

PG04-4 (20 points)

Give Q4 in the main function for this problem,

```
//Q4
cout << "[Q4]" << endl;
const CVec &g = b+c+a;
cout << "    ( g) " << b << "+" << c << "+" << a << "=" << g << endl;
CVec h(c+px);
cout << "    ( h) " << c << "+" << (*px) << "=" << h << endl;
```

Please complete two overloading operators (<< and +) to perform the insertion operations to a stream object and the append operation of two `CVec` objects by +. For example, for an object `x` with `vec` as `0X01`, **`4'b0X01`** will be shown on screen. Moreover, **`"0X01"+"11X"="0X0111X".`**

Hint: More than one overloading functions are needed for +. Read carefully!

The expected result is shown as

```
> ./pg04 4
[Q4]
   ( g) 5'b01X11+4'bX10X+2'b11=11'b01X11X10X11
   ( h) 4'bX10X+5'b01X11=9'bX10X01X11
>
```

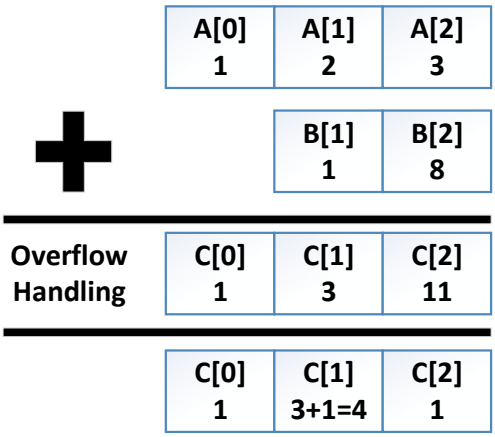
【PROBLEM SET 5】(30%)

Please design a class to implement Big-Integer operator. This class will be used to calculate the Fibonacci numbers. In this problem, the input is n, and the output is F(n), which F function is shown as follows:

F(n) = F(n-1) + F(n-2), and F(0) = 0, F(1) = 1

A simple class implementation has several steps. First, you'll divide the number into a sequence of array, like 123 (A array) and 13(B array) in Fig which is shown as below. Then, you should implement the operator overloading, operator+ and operator=. The last step is to show the number.

Here is an example for calculating 123 + 18, and the answer (C array) is 141.



Note that you cannot modify the main.cpp and the basic class **BigInt**, but you can add extra function or variable in your class if you want.

The main function is as the following:

```
int main() {
    BigInt fn_2(0), fn_1(1), fn;
    cin >> n;

    for (int i = 2; i < n; i++){
        fn = fn_1 + fn_2;
        fn_2 = fn_1;
        fn_1 = fn;
    }
    fn.print();
    return 0;
}
```

The basic class **BigInt** is as the following:

```
class BigInt {
private:
    int *arr;
public:
    BigInt();
    BigInt(int n);
    ~BigInt() { delete[] arr; }
    BigInt& operator =(const BigInt& other);
    BigInt operator +(const BigInt &other) const;
    void print() const;
};
```

The output should be

Case1

```
>./pg05
10
34
```

Case2

```
>./pg05
47
1836311903
```

Case3

```
>./pg05
93
7540113804746346429
```

Case4

```
>./pg05
150
616131474771527802958350162 6149
```