2015.2 Object-Oriented Programming and Design
Midterm Exam (Oct. 21th 7pm-8:20pm)

supervisor signature	

StudentID# : (   ) ,   Name : (   )

* You may answer in either Korean or English in problem 2~4. However, you should insert only English words in problem 1.

1. (18 points) Fill out the blanks (a)~(i) with the most appropriate English words.

- In C++, If we do not (a.   ) in a class, the compiler would generate a default constructor.
- Definition of software engineering: the establishment and use of sound engineering principles in order to (b.   ) obtain software that is reliable and works efficiently.
  One of the main goals of software engineering is to minimize (c.   ) of software development and maintenance.
- In the concept of information hiding, (d.   ) of an object can be modified only through (e.   ).
- Good SW design recognizes the inevitability of (f.   ) and plans an accommodation for them from the very beginning.
- Coupling is the degree to which the ability to fulfill a certain responsibility (g.   ) upon the actions of another component.
- **friend** function of a class has the right to access (h.   ) of the class while a typical function does not.
- (i.   ) is called when a new object is created and copied from an existing object with the same type.

2. (18points) List three main benefits of data encapsulation and explain for each benefit with sufficient details.

benefits (장점)	explanation for each benefit with sufficient details (각 장점에 대한 설명)
(i)	
(ii)	
(iii)	

3. (12 points) For a function **assert(e)**, fill out following blanks with appropriate words.
(1) If **e** is true, (   ). <----- describe what happens, if **e** is true.
(2) If **e** is false, (   ). <----- describe what happens, if **e** is false.
(3) What is the goal of using **assert(e)** function? (   )

4. (12 points)
(1) What is the meaning of "**static**" in line 2 of the following code? (You should explain the meaning of static member variable.)
(   )
(2) What is the lifetime of the static member variable **open_files**, meaning when is it created and deleted in memory?
    (i) Creation Time: (   ) , (ii) Deletion Time: (   )
(3) What is the lifetime of the member variable **var**, meaning when is it created and deleted in memory?
    (i) Creation Time: (   ) , (ii) Deletion Time: (   )

```
1: class File {
2:    static int open_files;         // static member variable of class File
3:    int var;
4:    // detail omitted
5: public :
6:    File(const String& filename, const String& modes);
7:    File(const File&);
8:    ~File();
9:    // detail omitted
10:}
```

5. (40 points) In our Project #2, class **inf_int** is defined as below. It can represent an almost infintely large integer number. (You may assume the maximum and minimum number it can represent are 10^{100} and -10^{100}). Insert your code in (1), (2), and (3). In (1), (2) and (3), Do not call other member functions. Instead, You may use C/C++ standard library functions in your code. Your code should manage the memory in a reasonable way. Your code should be grammatically and logically correct.

```
class inf_int
{ private:
         char* digits;            // points to a string of digits. Perform dynamic allocation when necessary.
         unsigned int length;     // stores the number of actual digits
         bool thesign;            // we assume that thesign is false if negative integer, true otherwise.
        // ex) 15311111111111111 : digits -> "11111111111111351", length=17, thesign=true;
        // ex) -12345555555555   : digits -> "55555555554321"    , length=14, thesign=false

 public :
         inf_int();                         // constructor. assign 0 as a default value.
         int_int(int n);                    // constructor. the input integer n is converted to inf_int format.
         inf_int(const inf_int& x);         // copy constructor

         friend bool operator==(const inf_int& , const inf_int&);
       // other member functions should be here but they are just not shown.
};
```

---

(1) Write C++ code for following function.

```
bool operator==(const inf_int& x, const inf_int& y)
{

}
```

(2) Write C++ code for following copy constructor.

```
inf_int::inf_int(const inf_int& x)
{

}
```

(3) Write C++ code for following constructor.

```
inf_int::inf_int(int n)
{

}
```