2016.2 Object-Oriented Programming and Design
Midterm Exam (Oct. 24th 7pm-8:20pm)

supervisor signature	

StudentID# : (   ) , Name : (   )

* You may answer in either Korean or English for explanation. 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.

- Coupling means the degree to which the ability to fulfill a certain responsibility (a.   ) the actions of another component. It is desirable to (b.   ) coupling.
- (c.   ) is an instance of a class.
- Using global variables can be dangerous because it can make all modules (d.   ) on one another.
- Goals of software engineering are
  (i) (e.   ),
  (ii) (f.   )
- (g.   ) is a programming and design technique that relies on the separation of interface and implementation.
  Interface is the outside or service view, describing (h.   ) an object does.
  Implementation is the inside view, describing (i.   ) an object does.

2. (26points) Answer to the following questions that are related to maintainability and data encapsulation.

(1) What is the definition of maintenance? Explain with sufficient details. (   )

(2) Fill out following blanks (a), (b), and (c) with appropriate words.
A program with high maintainability means that it is
- easy to (read/understand),
- easy to (a.   ),
- easy to (b.   ),
- easy to (c.   )

(3) Our OOP course emphasizes the importance of maintenance stage in software development process. Explain with sufficient details why the maintenance is so important.
(   )

(4) One of the main concepts in object oriented programming is data encapsulation. What is data encapsulation? Explain with sufficient details.
(   )

(5) Data encapsulation can make a program more understandable. Why? (   ).
(6) Data encapsulation can make a program safer. Why? (   )

3. (10 points) Implementation of SW system after design process can be divided into two subprocesses, (i) implementation of individual components and (ii) integration of components. For integration of components, it is often desirable that components are slowly integrated into completed system using stubs.

(1) Explain the meaning of 'slowly integrated'. Your answer should contain sufficient details.
(   )

(2) What is the meaning of 'stubs'? Explain.
(   )

4. (6 points) When deciding a name of a variable or a function, it is often suggested that the name should be evocative in the context of a problem. What is the meaning of 'evocative'? Explain.
(   )

5. (40 points) In our Project #2, class **inf_int** is defined as below. It can represent an almost arbitrarily large integer number in practice and the limit it can represent is maximized. 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 the sign is false if negative integer, true otherwise.
          // digits are stored in reverse order.
          // ex) 15311111111111111 : digits -> "11111111111111351", length=17, thesign=true;
          // ex) -12345555555555   : digits -> "55555555554321"    , length=14, thesign=false

 public:  inf_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)
{

}
```

(4) Generally speaking, we need to write a copy constructor in the class **inf_int**. Explain why?
( )

(5) What happens if we do not write a copy constructor in the class **inf_int**. Explain with sufficient details.
( )