

2016.2 Object-Oriented Programming and Design
Final Exam (Dec. 19th 7pm-8:20pm)

supervisor	
signature	

StudentID# : () , Name : ()

* You may answer in either Korean or English. As an exception, you can use only English words in problem 1.

1. (30 points) Complete following sentences about C++ by filling out blanks (a)~(o) with the most appropriate English words.

- (1) In C++, reference is (a).) for an object.
- (2) **friend** function of a class has the right to (b).).
- (3) Abstract class has at least one (c).).
- (4) (same as (c)) is declared with (d).).
This means we put the notation (same as (d)) in the declaration of (same as (c)).
- (5) The properties of abstract class:
 (i) We cannot create (e).) of an abstract class.
 (ii) An abstract class can only be used as (f).) class.
 (iii) Every derived class of an abstract class must (g).) the inherited (same as (c)) if (same as (e)) is to be created from it.
- (6) The purposes of an abstract class:
 (i) to provide (h).) and
 (ii) to delegate (i).) to the derived class.
- (7) Three basic components of STL are (j).), (k).), and (l).).
- (8) dynamic binding means that binding, the process of linking (m).) to (n).), occurs in (o).).

2. (8 points) Explain the main advantage and disadvantage of 'call-by-value' method by filling out following empty parentheses.

- (i) advantage: ()
- (ii) Explain why such advantage can be obtained: ()
- (iii) disadvantage: ()
- (iv) Explain why such disadvantage can be obtained: ()

3. (8 points) Answer to following questions.

- (1) In C++, what is the meaning of '**this**' keyword? ()
- (2) What is function overloading? Explain. ()

4. (8 points) Is following statement true or false? Choose the correct answer with circle-marking and explain.

(1) Inheritance is useful for increasing execution speed of an object-oriented program.

- Ans: (True / False),
If 'True', explain why inheritance is useful for increasing execution speed.
()

If 'False', explain why inheritance is not useful for increasing execution speed.
()

(2) In C++ STL, **sort(p, q)** means sorting all elements between **p** and **q** including **p** and **q**, when **p** and **q** are iterators.

- Ans: (True / False),
If 'True', explain average time complexity of **sort(p, q)** assuming the number of elements between **p** and **q** is **N**.
()

If 'False', explain which part in the above statement is wrong and how to correct the statement.
()

5. (15 points) Fill out empty box below with the output of following C++ code.

```
#include <iostream>
#include <stdio.h>

int main()
{
    int x = 3;
    int y = 7;
    int &z = y;
    int &u = x;
    int v = 11;
    u = v;
    v++;
    u=u-2;
    z=x;
    --x;
    y++;
    printf("x=%d, y=%d, z=%d, u=%d, v=%d\n",x,y,z,u,v);
    return 0;
}
```

Output :

6. (15 points) Read the C++ code, comments, and output of the following program and fill out empty boxes with appropriate code.

```
#include <stdio.h>

void square( [ ] )
{
    [ ]
}

int main()
{
    int x=5;
    square( [ ] ); // computes  $x^2$  and the result is assigned to x
    printf("x=%d\n",x); // x=25 will be printed because 25 is  $5^2$ 
    return 0;
}
```

Output :

x=25

7. (16 points) C++ code below shows generic stack implementation using template. Fill out empty boxes with appropriate codes.

```
template<typename T>
class Stack {
    int size;
    int top;
    T *stackPtr;
public:
    Stack(int n) { size=n; top=0; stackPtr=new T[size]; }
    ~Stack() { delete[] stackPtr; }
    bool push( [ ] ); // return true if push is successful
                      // return false if the stack is full
    bool pop( [ ] ); // return true if pop is successful
                      // return false if the stack is empty
    bool isEmpty() { if (top<=0) return true; else return false; }
    bool isFull() { if (top>=size) return true; else return false; }
};

// Insert your code for push member function
```

```
#include <iostream>
int main()
{
    int x, y;
    float xf, yf;
    Stack<int> s1(5);
    Stack<float> s2(5);
    s1.push(5); s1.push(8);
    s1.pop(x); s1.pop(y);
    s2.push(5.3); s2.push(8.1);
    s2.pop(xf); s2.pop(yf);
    std::cout << x << " " << y << std::endl;
    std::cout << xf << " " << yf << std::endl;

    return 0;
}
```

Output :

8 5
8.1 5.3