2015.2 Object-Oriented Programming and Design
Final Exam (Dec. 16th 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. (22 points) Complete following sentences by filling out blanks (a)~(k) with the most appropriate English words.
You can use only English words in this problem 1. Otherwise, you will get some penalty.

(1) In STL, a container manages storage space for its elements and provides member functions to access them, either directly or through (a.).

(2) Just like arrays, STL vectors use (b.) storage locations for their elements, which means that their elements can also be accessed using offsets on regular pointers to its elements, and just as efficiently as in arrays. But unlike arrays, their size can change (c.), with their storage being handled automatically by the container.

(3) In C++, (d.) is a member function that you expect to be redefined in derived classes. When you refer to a derived class object using a pointer or a reference to the base class, you can call (same as (d)) for that object and execute the derived class's version of the function..

(4) UML uses (e.) notation: clearer than natural language and code.
UML is not (f.) any one language or technology.

(5) A vector's member function (g.) returns the size of the storage space currently allocated for the vector, expressed in terms of elements.
A vector's member function (h.) returns theoretical limit on the size of the vector.

(6)

```
class lifeform
{ public: virtual void die()=0; };
```

In above code, '**=0**' means **die()** function is (i.) function.
This also means that the class **lifeform** is (j.) class.

(7) "dynamic binding" means that
binding (the process of linking procedure call to a specific sequence of code) occurs in (k.).

2. (18 points) There are differences among following three function parameter types (a), (b), and (c). **X** is a class name and **T** is a data type.

(a) `void X::f(T arg)`
(b) `void X::f(T* argp)`
(c) `void X::f(const T* argp)`

(1) What are the purposes of using * (passing address) in (b) compared to using parameter type (a)? There are two important purposes (benefits). Explain.

(purpose 1:)

(purpose 2:)

(2) what is the purpose (benefit) of using **const** in (c) compared to using parameter type (b)? Explain.
( )

3. (12 points)
(1) What is the meaning of '**protected**' access (visibility) modifier? Explain.
( )

(2) Using '**protected**' instead of '**private**e' means we have less ( ). Therefore, it is desirable to use '**protected**' only where it is really necessary.
Fill out above blank with the most appropriate word(s).

4. (12 points) Is following statement true or false? Choose the correct answer with circle-marking and explain.
(1) Template mechanism is particularly useful for defining container classses.
Ans: (True / False),
If 'True', explain why template mechanism is useful for defining container classes.
( )

If 'False', explain why template mechanism is not useful for defining container classes.
( )

(2) One of the main advantage of dynamic binding is increased execution speed.
Ans: (True / False),
If 'True', explain why using dynamic binding can increase execution speed.
( )
If 'False', explain why increasing execution speed is not advantage of dynamic binding.
( )

5. (18 points) Write a C++ function "**sum**" that computes and returns summation of input array. Note that the input array type is a generic type, meaning the type can be **int** or **float**. Therefore, you must use template to write the "**sum**" function that can accept **int** type array and **float** type array, as shown in the following sample code and its input/output result.

```
#include <iostream>
using namespace std;

// Write your sum function here using template.
// sum takes two parameters (i) input array, (ii) array size
```

```
int main()
{
    float a[5];
    int b[5];
    int i;
    for (i=0;i<5;i++) cin >> a[i];
    for (i=0;i<5;i++) cin >> b[i];
    cout << sum(a,5) <<  ","  << sum(b,5) << endl;
    return 0;
}
```

```
input :
1.1 2.2 3.3 4.4 5.5 1 2 3 4 5
output :
16.5,15
```

6. (18 points) What is the output of the following C++ program to the screen?

```
#include <iostream>
using namespace std;

class B {
public:
  B() { z=-5;  cout << "B(): z=" << z << endl; }
  B(int z_val):z(z_val) { cout << "z=" << z << endl;  z++;}
  virtual int get_val() { --z; return (z-1); };
  int gv2() { --z; return (z-2); }
private :
  int z;
};

class D1 : public B {
public:
  D1() { x=7; cout << "D1(): x=" << x << endl; x++; }
  D1(int x_val): x(x_val) { cout << "x=" << x << endl; x--;}
  virtual int get_val() { x++; return x; };
  int gv2() { x++; return x+1; }
private:
  int x;
};

class D2 : public B {
public:
  D2() { y=2; cout << "D2(): y=" << y << endl; }
  D2(int y_val): y(y_val) { cout << "y=" << y << endl; }
  int get_val() { y--;  return y; }
  virtual int gv2() { y--; return y*y; };
private:
  int y;
};

void myf1(B& f) { cout << "7 : " << f.get_val() << endl ; }
void myf2(B& f) { cout << "8 : " << f.gv2() << endl ; }
void myf3(D1& f) { cout << "9 : " << f.get_val() << endl ; }
void myf4(D1& f) { cout << "10 : " << f.gv2() << endl ; }
```

```
int main()
{
  B Zero(0);   D1 Two;   D2*  d2ptr;
  B* B_ptrArray[2];
  B_ptrArray[0] = &Zero;
  B_ptrArray[1] = &Two;
  d2ptr = new D2 ;

  cout << "0 : " << B_ptrArray[0]->get_val() << endl;
  cout << "1 : " << Two.get_val() << endl;
  cout << "2 : " << Two.gv2() << endl;
  cout << "3 : " << B_ptrArray[1]->get_val() << endl;
  cout << "4 : " << B_ptrArray[1]->gv2() << endl;
  cout << "5 : " << d2ptr->gv2() << endl;
  cout << "6 : " << d2ptr->get_val() << endl;
  myf1(Two);
  myf2(Two);
  myf3(Two);
  myf4(Two);
  delete d2ptr;
  return 0;
}
```

```
Output : (PUT YOUR ANSWER HERE)
```