CS 250 Exam 3 Review

Vocabulary: Inheritance, superclass, subclass, base class, derived class, is-a, has-a, composition, overloading, access specifiers: public vs private vs protected, this, *this, regular variable versus pointer variable (int versus int *, char versus char *, …), sizeof, address operator &, dereference operator *, ->, pointer operations (pInt + 2, *(pInt + 2), ++pInt, pInt++…), difference between arrays and pointers, writing code using array notation versus pointer notation, pointers as function arguments, int * versus const int * versus int * const, destructor, polymorphism, virtual functions, static variables, UML Diagrams

1. Show the output of the following program:

```
class Base {
  public:
     Base (){cout << "Base" << endl;}
    Base (int i){ cout << "Base" << i << endl;}
    ~Base(){cout << "Destruct Base" << endl;}
};

class Der: public Base {
  public:
    Der () {cout << "Der" << endl;}
    Der (int i): Base(i) { cout << "Der" << i << endl;}
    ~Der () {cout << "Destruct Der" << endl;}
};

int main() {
  Base a;
  Der d(2);
  return 0;
}
```

2. A Cube is derived from a Rectangle. The Rectangle and Cube are to have appropriate constructors. The Rectangle is to have an area function and a Cube is to have a volume function. First, write the UML diagram that shows this inheritance and then write the C++ code to declare & define each class.
3. Time can be displayed in Regular Time or Military Time. Examples:
   Military Time is 22:00:00
   Regular Time is 10:00:00 PM
   Write the proper UML diagram for MilitaryTime and RegularTime.

Regular Time	Military Time	Regular Time	Military Time
12:00 a.m.	0000	12:00 p.m.	1200
1:00 a.m.	0100	1:00 p.m.	1300
2:00 a.m.	0200	2:00 p.m.	1400

3:00 a.m.	0300	3:00 p.m.	1500
4:00 a.m.	0400	4:00 p.m.	1600
5:00 a.m.	0500	5:00 p.m.	1700
6:00 a.m.	0600	6:00 p.m.	1800
7:00 a.m.	0700	7:00 p.m.	1900
8:00 a.m.	0800	8:00 p.m.	2000
9:00 a.m.	0900	9:00 p.m.	2100
10:00 a.m.	1000	10:00 p.m.	2200
11:00 a.m.	1100	11:00 p.m.	2300

Destructor

- What is a destructor and when is a destructor called?
- Why would you want to have a destructor?
- When is a virtual destructor necessary?

Pointers & dynamic memory allocation

- Consider the following C++ program:

```
class Base {
 public:
 void show() {
  cout << "Base class";
 }
};

class Derived:public Base {
 public:
 void show(){
  cout << "Derived Class";
 }
}

int main() {
 Base* b;
 Derived d;
 b = &d;
 b->show();
}
```

- Does the above example contain any static binding? If so, what?
- Does the above example contain any dynamic binding? If so, what?