

REGENT UNIVERSITY
COLLEGE OF SCIENCE AND TECHNOLOGY



EXAMINATION PAPER

RESIT EXAMINATIONS,
2ND JULY 2010

COURSE: OBJECT ORIENTED PROGRAMMING
USING C++

COURSE CODE: SICS 1523

TIME: TWO HOURS

LECTURER: KENNETH K. AZUMAH

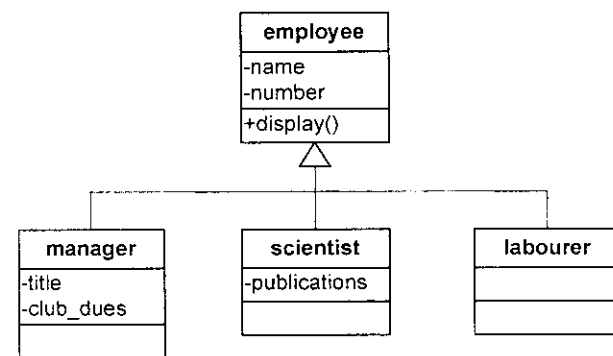
Please Read ALL Instructions

Section A [20 marks]

For each question, circle all answers that apply.

(Recommended Time: 20 minutes)

Use the diagram below in answering questions 1 – 4



1. The class structure above demonstrates

- a. inheritance.
- b. composition.
- c. classes.
- d. the "kind of" relationship.
- e. reusability.

2. In the diagram above, the _____ class is _____ the _____ class(es).

- a. scientist, the base class for, employee
- b. labourer, derived from, employee
- c. employee, composed of, manager, scientist, and labourer

- d. employee, derived from, labourer
- e. labourer, the same as, employee

3. An object of the scientist class contains instance data representing

- a. the employee name, employee number, and number of publications.
- b. only the number of publications.
- c. only the employee name and employee number.
- d. the title and golf club dues.
- e. the location of the object.

4. From the manager class, you call the display() function in employee to

- a. display all the data on a manager.
- b. satisfy the formal requirements of the compiler.
- c. display the manager's name and number.
- d. display the manager's title and golf club dues.
- e. let the compiler know you're dealing with an object derived from the employee class.

5. An abstract class is

- a. one whose objects are identical to those of some derived class.
- b. one from which no objects will be instantiated.
- c. one that contains common elements of derived classes.
- d. any base class.
- e. any derived class.

6. Which of the following are true?

- a. A derived class constructor is executed before the base class constructor.

- b. A derived class constructor is executed after the base class constructor.
- c. A derived class destructor is executed before the base class destructor.
- d. A derived class destructor is executed after the base class destructor.
- e. Derived and base class constructors are executed simultaneously.

7. For a derived class constructor with arguments to call a base class constructor, it must
- a. make the call in the usual way from within its function body.
 - b. finish executing before calling the base class constructor.
 - c. use any arguments to itself in the call to the base class constructor.
 - d. place the call to the base class constructor on its initialization list.
 - e. make no explicit call, because the system will handle it automatically.

8. If there's a constructor with arguments in a derived class, then you must have
- a. at least a no-argument constructor in the base class.
 - b. a constructor with the same number of arguments in the base class.
 - c. a no-argument derived class constructor (assuming you'll instantiate objects without arguments).
 - d. a no-argument base class constructor (assuming you'll instantiate objects without arguments).
 - e. instantiated objects of the base class.

9. The initializer list in a constructor typically
- a. initializes variables of basic types in its own class.
 - b. calls a constructor in a base class.
 - c. initializes variables of basic types in a base class.
 - d. calls other constructors in its own class.
 - e. assigns values to existing variables.

10. Inheritance facilitates reusability because
- a. child objects cannot be modified.
 - b. the base class need not be modified to derive a new class.
 - c. programming objects are more like real-world objects.
 - d. objects of the base class can be treated as objects of the derived class.
 - e. derived class objects inherit only the desirable features of the base class.

SECTION B [40 marks max]

Answer all questions in this section

Recommended time: 80 mins

- 1) Study the UML static class diagram below and answer the following questions:
 - a) Convert the diagram into C++ code bearing in mind the following hints
 - i. Let each constructor initialize its class. Choose your own initial values for the attributes.

[6 mks]

ii. The accessor and mutator operations (functions) in **Employee** should be implemented fully.

[10 mks]

iii. Implement accessor and mutator functions for the Lecturer and Technician classes

[4 mks]

iv. The operations (functions) in **Lecturer** and **Technician** may each have an empty function body.

[6 mks]

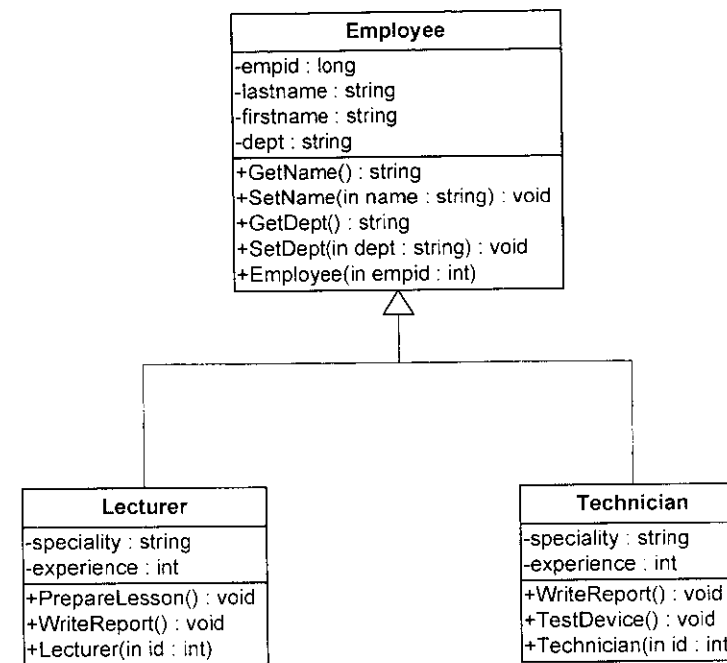
b)

i. Which operation(s) in the **Lecturer** and **Technician** classes can be abstracted into the **Employee** class?

[2 mk]

ii. What object-oriented property is being exhibited in the diagram? Explain.

[2 mks]



2) Operator Overloading [10 mks]

The **Airtime** object has hour and minute properties. Two **Airtime** objects can be added together to obtain another **Airtime** object. If the sum of the minute properties of the two objects exceeds 60, the hour should be increased by 1. (eg. [5 hours, 57min] + [5hours, 10min] = [11 hours, 7min].)

Consider the **Airtime** class below...

```
class Airtime{
public:
int hour;
int min;
Airtime(){
    hour=0;
    min=0;
}
Airtime(int x, int y){
    hour = x;
    min = y;
}
Airtime Add(Airtime airtime){
    Airtime c;
    c.hour = this->hour + airtime.hour;
    c.min = this->min + airtime.min;
    return c;
}
};
```

Code Listing A

Use Code Listing A above to answer to following question:

In the Complex class above two Airtime objects C and D can be summed using the **Add** function thus:

```
Airtime Sum = C.Add(D);
```

Rewrite the **Add** function by overloading the '+' operator so that the two complex numbers can be summed using:

```
Airtime Sum = C + D;
```