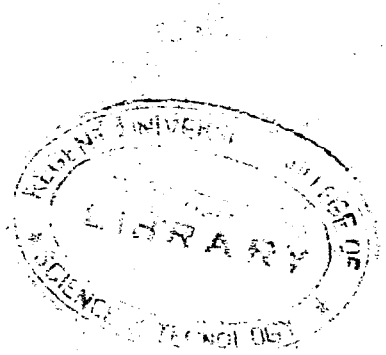


REGENT UNIVERSITY

COLLEGE OF SCIENCE AND TECHNOLOGY

SCHOOL INFORMATICS AND ENGINEERING

EXAMINATION PAPER

END OF SECOND SEMESTER EXAMINATIONS

SICS 1523: OBJECT-ORIENTED PROGRAMMING USING C++

DATE: MAY 2009

TIME ALLOWED: THREE HOURS

EXAMINATION MATERIAL PROVIDED: NONE

INSTRUCTION: ATTEMPT ALL QUESTIONS

LECTURER: KENNETH K. AZUMAH

Attempt All Questions

Question 1 **[22 marks max]**

Carefully study Code Listing 1 below and answer the questions that follow.

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

class Mammal{
public:
        virtual void Identify(){
        cout << "I am a mammal" << endl;
        }
        Mammal(): age(0){}
        Mammal(int age){this->age=age;}
protected:
        int age;
};
class Cat: public Mammal{
public:
        void Identify(){
          cout << "I am a Cat" << endl;
        }
        Cat(){}
        Cat(int a){ this->age = a; }
        int getAge(){ return age;}
};
class Leopard: public Cat{
public:
        Leopard(){}
        void Identify(){
          cout << "I am a Leopard" << endl;
        }
};
```

```
P1.     int main(){
P2.        Mammal * mammal[2];
P3.        Cat c = Cat(5);
P4.        mammal[0] = &c;
P5.        mammal[0]->Identify();
P6.        Leopard leo = Leopard();
P7.        mammal[1] = &leo;
P8.        mammal[1]->Identify();
P9.        cout << c.getAge() << endl;
P10.       cout << leo.getAge() << endl;
P11.       getchar();
P12.       return 0;
P13.    }
```

Code Listing 1

a. Write out the output generated by the code listing.
[8 marks]

b. How will the output change:

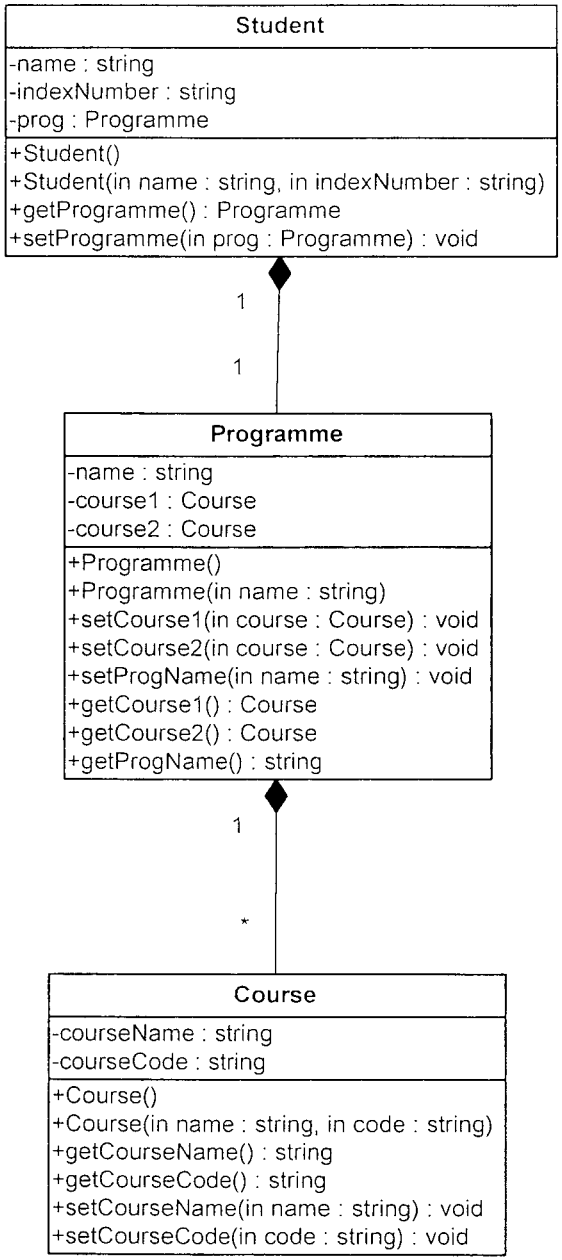
i. If line P3. in the main() function above was changed to
`Cat c = Cat();`
[3 marks]

ii. If the **Mammal** class was modified to

```
class Mammal{
      public:
         void Identify(){
            cout << "I am a mammal" << endl;
         }
         Mammal(): age(0){}
         Mammal(int age){this->age=age;}
      protected:
            int age;
};
```

[4 marks]

c. Draw the UML diagram that can represent the classes and the relationship(s) between them.
[7 marks]



Question 2 **[23 marks max]**

Convert the following UML diagram into C++ code. Implement fully all constructors providing your own initial values for the class data members (attributes) in the default constructors. Implement fully all accessor- and mutator-functions. Your implementation should enable the following programme snippet to work.
[1 mark per function/constructor implemented correctly]

```
#include <iostream>
using std::cout;

int main(){
    Course k1 = Course("Object Oriented Prog", "SICS 1523");
    Course k2 = Course("Principles of Prog", "SICS 1513");

    Programme p = Programme("Computer Science");
    p.setCourse1(k1);
    p.setCourse2(k2);

    Student s = Student("John", "0123401");
    cout<<s.getProgramme().getCourse1().getCourseCode()<<endl;
    s.setProgramme(p);

    cout<<s.getProgramme().getCourse1().getCourseCode()<<endl;
    cout<<s.getProgramme().getCourse2().getCourseCode()<<endl;

    getchar();
    return 0;
}
```

Question 3 **[15 marks max]**

```
class Complex{
      public:
      float real;
      float imaginary;
      Complex(){
         real=0.0;
         imaginary(0.0)
      }
      Complex(float x, float y){
        real = x;
        imaginary = y;
      }
      Complex Add(Complex complex){
        Complex c;
        c.real = this->real + complex.real;
        c.imaginary = imaginary + complex.imaginary;
        return c;
      }
};
```

Code Listing 2

Use Code Listing 2 to answer to following questions:

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

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

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

`Complex Sum = C + D;`

[4 marks]

b. Add a multiplication functionality to the Complex class by overloading the '*' operator. This should facilitate the multiplication of two complex objects thus:

`Complex Sum = C * D;`

`Hint: (a + ib)*(x + iy) = (ax - by)+ i(ay + bx)`

[11 marks]