UNIVERSITY OF LONDON 291 0212 ZA

FOR EXTERNAL STUDENTS (WEST)

B.Sc. Examination 2008
COMPUTING AND INFORMATION SYSTEMS AND CREATIVE COMPUTING
2910212 Programming: Advanced Topics and Techniques

Duration: 3 hours

Date and time: Thursday 8 May 2008 : 10.00 – 1.00 pm

Answer SIX questions.
Full marks will be awarded for complete answers to SIX questions.

You must answer THREE questions from section A and THREE questions from section B.
In section B you must answer at least ONE question on Prolog (questions 9 and 10).

There are 150 marks available on this paper.

A hand held calculator may be used when answering questions on this paper but it must not be pre-programmed or able to display graphics, text or algebraic equations. The make and type of machine must be stated clearly on the front cover of the answer book.

THIS EXAMINATION PAPER MUST NOT BE REMOVED FROM THE EXAMINATION ROOM

SECTION A

Question 1

The class *Triangle* makes an object with three int datafields.

```
public class Triangle
{
     protected int x,y,z;

     public Triangle(int newX, int newY, int newZ)
     {
          x = newX;
          y = newY;
          z = newZ;
     }

     public Triangle(int anotherX, int anotherY, int anotherZ)
     {
          x = anotherX;
          y = anotherY;
          z = anotherZ;
     }

     public int getX()
     {
          return x;
     }

     public void setX(int newX)
     {
          x = newX;
     }

     public int getY()
     {
          return y;
     }

     public void setY(int newY)
     {
          y = newY;
     }
}
```

(a) Does *Triangle* have accessor methods? If so identify them. [2 marks]

(b) What is the purpose of accessor methods? [2 marks]

(question continues on next page)

(c) Write a getter method and a setter method for the *z* variable. [4 marks]

(d) How many constructors does *Triangle* have? [1 mark]

(e) Will the *Triangle* class compile? Give a reason for your answer. [3 marks]

(f) Give the signatures of four additional legitimate constructors that the *Triangle* class could have. [2 marks]

(g) Implement one example of each of the other possible constructors for *Triangle* that you have identified. [8 marks]

(h) Why are constructors static? [3 marks]

Question 2

(a) Do design patterns primarily involve the re-use of code or of ideas? Justify your answer. [4 marks]

(b) The following class has been written to implement a particular design pattern in Java.

```
abstract class Induction
{
     void inductionPack()
     {
          System.out.println ("please collect your
          induction pack on arrival from reception");
     }

     void history()
     {
          System.out.println ("please read the book on
          company history in your induction pack");
     }

     void healthSafety()
     {
          System.out.println ("please read the health
          and safety information carefully");
     }

     abstract void reportToRoom();
}
```

(i) Name the design pattern and explain how it is implemented in Java, making reference to the *Induction* class as an example. [4 marks]

(ii) Employees of the type janitor have to report to room B110. Write a subclass *Janitor* that implements the abstract method reportToRoom. [4 marks]

(iii) Write a main method within the class *Janitor* that correctly calls all the methods of the class and the superclass. [5 marks]

(c) Name two differences between an abstract class and an interface. [4 marks]

(d) Why would an interface not be suitable for implementing the template design pattern? [4 marks]

Question 3

(a) Explain each of the following:

(i) inheritance for extension; [3 marks]

(ii) inheritance for specialization; [3 marks]

(iii) inheritance for specification. [3 marks]

(b) The abstract class *Vegetable* has one abstract method, *seeds*.

```
abstract class Vegetable
{
     boolean edible;
     boolean plant;
     String colour;

     public Vegetable()
     {

          edible = true;
          plant = true;
     }

     void setColour (String newColour)
     {
          colour = newColour;
     }

     String getColour()
     {
          return colour;
     }

     abstract boolean seeds();
}
```

(i) Extend the abstract class *Vegetable* to a *Squash* subclass that implements the boolean *seeds* method (you do not need to write a constructor for the subclass). [4 marks]

(ii) Does the *Squash* subclass demonstrate inheritance for specification or for extension? Justify your answer. [2 marks]

(c) Define method overloading. [3 marks]

(d) Write a static method *add* that adds together two integers. [3 marks]

(e) Demonstrate overloading by writing a suitable second *add* method. [2 marks]

(f) What is the difference between overriding and overloading in terms of method signatures? [2 marks]

Question 4

(a) The *swap* method swaps two items in a given array.

```
public static void swap(int[]a, int i, int j)
{
      // REQUIRES: 0 <= i,j < a.length
      // EFFECTS: Swaps the contents of a[i] and a[j]
      // MODIFIES: a
}
```

(i) What is the purpose of the three comments at the start of the method? [2 marks]

(ii) In general terms what does the REQUIRES comment document? [2 marks]

(iii) In general terms what does the EFFECTS comment document? [2 marks]

(iv) Implement the *swap* method. [6 marks]

(b) I intend to make an object with two int datafields. Before I make the object I will write a method *fact* to return the factorial of *x* so that I can use *x* as one datafield, and *x!* as the other. Should my method be a static or an instance method? [2 marks]

Give a reason for your answer. [2 marks]

(c) Write the *fact* method as a **recursive** method (no credit will be given for iterative methods). [9 marks]

NB: The factorial of *x* is written *x!* and is defined to be $x(x\text{-}1)(x\text{-}2)\ldots 2 \times 1$.
Therefore $4! = 4 \times 3 \times 2 \times 1$

Question 5

Implement a class to do the following (you may answer all questions within one class if you wish):

(a) Display a rectangle in a 400 x 400 JFrame; [6 marks]

(b) Fill the rectangle with the colour red; [5 marks]

(c) Place a button with no functionality into the JFrame; [4 marks]

(d) Add the following functionality to the button: when it is pressed the rectangle disappears. [10 marks]

SECTION B

Q6.

a) In SML there are a number of *comparison operators* (< for example) and three logical operators: *andalso*, *orelse* and *not*.
For each of the four terms in italics explain their use and give an example.

[8]

b) Amongst others, SML has the following primitive types of values: *reals*, *boolean*, and *strings*.
Explain the meaning of each term in italics and give an example as well as a use of each.

[6]

c) Give a step by step evaluation of:
If 7 > 3 then if 1+10=11 then 9+2 else 2-3 else ~2+3;

[4]

d) Explain the Let statement in SML giving its syntax, its uses and suitable examples.

[7]

Q7.

a) Distinguish between the SML data types ***Lists***, ***Records*** and ***Tuples*** giving an example and typical use of each.

[5]

b) Write SML expressions to extract:
   i) The element in position 3 in a tuple, so that c is extracted from (a, b, c, ...)
   ii) The second element from a list so that 'second' is extracted from ["first", "second", ...]

[4]

c) Explain the mechanisms allowing us to obtain parts from an SML record.

[4]

d) Using the example of a simple shopping list, describe the mechanism SML provides for user defined data types.

[5]

e) Using your definitions from d) above, outline algorithms for adding, removing and looking up items in such a list.

[7]

Q8.

Using a named procedural language of your choice, SML (as an example of a functional language) and Prolog (as an example of a logic programming language) compare and contrast these three programming styles in terms of:

a) How variables are used

b) The ways that we think of program execution

c) Overloading and polymorphism

In each case a), b) and c) illustrate your answers with examples.

[25]

Q9. Prolog

a) Describe the use of *not* in Prolog, giving a suitable example to illustrate your answer.

[2]

b) Consider the following Prolog rules:

```
member(X, [X|T]).
member(X, [_|T]) :- not member(X, T).
```

i) Give a step by step trace of member(1, [1,2]).

ii) Give a step by step trace of member(2, [1, 2, 3]).

[5]

c) Given the predicate *member* in a) above:

i) What output would the code above for *member* give to the query member(X, [1,2,3]).

ii) What would be the response if a sequence of semicolons ';' each followed by return were typed in response to the result of i) above?

Give reasons for your answers.

[4]

d) Explain the difference, if any, that would be made to your answer in c) above had the not been omitted so that the second line was:

t member(X, [_|T]) :- member(X, T).

Explain your reasoning. [6]

e) Define the terms *functor* and *arity* as used in Prolog giving a suitable example. [4]

f) Explain the use of the *dot functor* in Prolog to represent lists. [2]

g) Give the dot form of the following lists: [2]

Q10. Prolog

a) Write the following Prolog predicates:

i) len(L, N) that takes a list L and returns the length of the list as N so that len([1, 3, 2], N). would return with N=3.

[2]

ii) sum(L, S), that takes a list of numbers L and returns with S containing the sum of those numbers.

[3]

iii) odd(L, Od), that takes a list of integers and adds the odd numbers in the list, so that odd([1,2,3,4,5,6], Od). Results in: Od = 9

[3]

iv) even(L, E), that takes a list of integers and adds the even numbers in the list, so that even([1,3,2,4,3,6], E) results in E = 12.

[3]

v) less(L, P, R) which takes a list, L, of numbers and a number P and returns the list of numbers in L that are smaller than P. So that, for example, less([1, 5, 3, 4, 2], 2.5, R) results in R= [1, 2].

[3]

b) The course guide says: 'The data objects of Prolog are known as terms. There are three distinct types of terms: *constants*, *variables* and *structures*.'

i) Define the three terms in italics in the quote above and give an example of each.

[6]

ii) Explain how binary trees can be represented in Prolog, giving appropriate representations of 3 example binary trees.

[5]

END OF EXAMINATION