UNIVERSITY OF LONDON 291 0212 ZB

FOR EXTERNAL STUDENTS (EAST)

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 *Point* makes an object with two int datafields.

```
public class Point
{
     protected int x,y;


     public Point(int newX, int newY)
     {
          x = newX;
          y = newY;
     }


     public Point(int anotherX, int anotherY)
     {
               x = anotherX;
               y = anotherY;
     }


     public int getX()
     {
          return x;
     }


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

(a) Does *Point* have a getter method? If so identify it. [2 marks]

(b) Does *Point* have a setter method? If so identify it. [2 marks]

(c) What is the purpose of getter and setter methods? [4 marks]

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

(e) What is the general name for getter and setter methods? [1 mark]

(question continues on next page)

(f) How many constructors does *Point* have? [1 mark]

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

(h) Give the signatures of three additional legitimate constructors that the *Point* class could have. [2 marks]

(i) Implement each of the three possible constructors for *Point* that you have identified. [6 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 used in the class *Induction* and describe it in general terms. [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]

(iv) Write the shortest possible constructor for *Janitor.* [4 marks]

(v) What would happen if you tried to compile the subclass *Janitor* without a constructor? Justify your answer. [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) Consider the following class, *Vegetable*:

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

        public Vegetable()
        {
              edible = true;
              plant = true;
        }

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

        String getColour()
        {
              return colour;
        }

        boolean seeds()
        {
              return false;
        }
}
```

(i) Using the *Vegetable* class write a *greenVeg* subclass with a method *Colour* that prints the colour of the object to the screen (you do not need to write a constructor for the subclass). [4 marks]

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

(iii) Using the *Vegetable* class write a subclass *Tomato* with a

(question continues on next page)

boolean *seeds* method that takes no parameters and returns true. [4 marks]
There is no need to write a constructor for the subclass.

(iv) Does the subclass *Tomato* demonstrate inheritance for specialisation or for extension? Justify your answer. [2 marks]

(c) Define method overriding. [3 marks]

(d) Does the *seed* method in the subclass *Tomato* override or overload the seed method in its parent class *Vegetable*? [1 mark]

Question 4

(a) The *swap* method swaps two items in a given array. It is documented by its REQUIRES, EFFECTS and MODIFIES comments.

```
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) Why is it good practice to document code with REQUIRES, EFFECTS and MODIFIES comments? [2 marks]

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

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

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

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

Give a reason for your answer. [2 marks]

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

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 colour of the rectangle changes to blue. [10 marks]

SECTION B

Q6.

a) "At its simplest level, an SML program consists of an *expression*. In turn, these expressions consist of *operations*(or operators), *operands*, and *punctuation marks*."
Define each of the terms in italics in this quoted statement and give an example of each.

[8]

b) Amongst others, SML has the following primitive types of values: *integer*, *boolean* and *characters*. 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 1 < 3 then if 1+1=2 then 9*2 else 2+3 else 2-3;

[4]

d) What is an *exception* in SML? Using a function of two real parameters that divides the first by the second as an example, show how exceptions are raised and handled in SML.

[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 list, so that c is extracted from [a, b, c, …]
ii) The second element from a tuple 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 telephone directory, 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 directory.

[7]

Q8.

Using a 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 parameters are passed

b) The scope of variables

c) How user defined datatypes are represented

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

[25]

Q9. Prolog

a) Describe the use of the *cut* in Prolog, giving a suitable example to illustrate your answer.

[2]

b) Consider the following Prolog rules:

```
member(X, [X|T]).
member(X, [_|T]) :- 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 a cut been included before the end of the second line of member (so that the line read: member(X, [_|T]) :- member(X, T),!). Explain your reasoning.

[6]

e) Explain the use of assert and retract(and their other forms) in Prolog giving examples to illustrate your explanation. [8]

Q10.

a) Explain the meaning and use of the reserved word *is* in Prolog. [2]

b) List the operators that are used to compare numbers in Prolog. [3]

c) Write a Prolog predicate *choose(L, N, E)* which results in *E* being the element of list *L* in position N, so that choose([5, 1, 2, 3, 4, 5, 7], 4, X) results in X=3.

[4]

a) Write a Prolog predicate from_to(L, N, M, R) which given list L and two integers N and M results in R being the list from its Nth to its Mth elements. So for example from_to([a,b,c,d,e,f,g,h,i], 3, 6, L). results in L=[c,d,e,f]. For each clause in your answer give a few lines of text explaining how it works.

[16]

END OF EXAMINATION