

THIS PAPER IS NOT TO BE REMOVED FROM THE EXAMINATION HALLS

UNIVERSITY OF LONDON

291 0212 ZA

**BSc Examination**  
for External Students

**COMPUTING AND INFORMATION SYSTEMS AND  
CREATIVE COMPUTING**

**Programming: Advanced Topics and Techniques**

Dateline: Monday 11 May 2009 : 2.30 – 5.30 pm

Duration: 3 hours

Candidates should answer **SIX** questions. Full marks will be awarded for complete answers to **SIX** questions.

Candidates must answer **THREE** questions from **Section A** and **THREE** questions from **Section B**. In Section B you must answer **ONE** question on Prolog (questions 9 and 10).

There are 150 marks on this paper.

Calculators may NOT be used.

© University of London 2009

## SECTION A

### Question 1

- (a) Which keyword is used when exceptions are declared in Java? [1 mark]
- (b) Explain how exceptions are used for handling errors, such as incorrect input by the user, using the *absValue()* method in the *AbsVal* class below as an example. [6 marks]

```
public class AbsVal
{
    static int  onlyPositive (int x) throws NewException
    {
        if (x<0) throw new NewException();
        return x;
    }

    static int absValue (int x)
    {
        try
        {
            return onlyPositive (x);
        }
        catch (NewException e)
        {
            return (-x);
        }
    }

    public static class NewException extends
    RuntimeException
    {
        public NewException (String msg)
        {
            super (msg);
        }
        public NewException() {}
    }
} // end of class
```

- (c) Write a main method that calls the *absValue()* method with -3 and assigns the value returned to an int variable *x*, and that next calls the *absValue()* method with 3, assigning the value returned to an int variable *y*. The values of *x* and *y* are then printed to the screen on separate lines. [4 marks]

(question continues on next page)

- (i) What will be output to the screen when the programme is run? [2 marks]
- (ii) What would happen if you compiled your programme with the statement `absValue(3.0)` in your main method? [4 marks]
- (d) Consider the recursive method `power()` below, which is intended to return  $a$  raised to the power of  $b$ , for example if `power(2, 3)` is entered then it is intended that the method will return  $2 \times 2 \times 2 = 8$ . If a negative number is entered it is intended that the method will throw an Exception (assume that the exception has been defined).

```
public int power (int a, int b) throws
NegativeException
{
    if (b<0) throw new NegativeException
    ("positive numbers only");
    else return a*power(a,b-1);
}
```

- (i) Explain why the method will not do what it is intended to do [6 marks]
- (ii) Explain how the method could be corrected [1 mark]
- (iii) Write the corrected the code so that the method will work as intended. [1mark]

## Question 2

- (a) Explain how design patterns help us to re-use ideas [4 marks]
- (b) Define the problem that the composite design pattern helps to solve [4 marks]
- (c) Explain briefly how Java implements the iterator design pattern [4 marks]
- (i) Name the design pattern used in the class *Induction* and describe it in general terms. [4 marks]

```
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();
}
```

- (ii) Define a concrete sub-class *Clerk* that has to report to room B110, and another concrete sub-class *Manager* that has to report to the Boardroom. The *Manager* class has an additional method *CompanyCar()* that prints a message telling the employee to collect their car keys from reception on arrival. [9 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 abstract class, *Mammal*, which has one abstract method, *reproduce()*

```
abstract class Mammal
{
    boolean spine;
    boolean warmBlooded;

    public Mammal()
    {
        spine = true;
        warmBlooded = true;
    }

    boolean hasFur()
    {
        return true;
    }

    abstract String reproduce();
}
```

(i) Extend the abstract class *Mammal* to a *duckBilledPlatypus* subclass that implements the string *reproduce* method by returning the sentence: "This mammal lays eggs."

*NOTE: you do not need to write a constructor for the subclass* [4 marks]

(ii) Does the *duckBilledPlatypus* subclass demonstrate inheritance for specification or for extension? Justify your answer.

[4 marks]

(iii) Extend the abstract class *Mammal* to a *human* subclass that implements the string *reproduce* method by printing out the sentence: "This mammal gives birth to live young." and that has its own *hasFur()* method that returns false.

[4 marks]

*NOTE: you do not need to write a constructor for the subclass*

(iv) Does the *human* subclass demonstrate inheritance for specification or specialization or both? Justify your answer.

[4 marks]

#### 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
}
```

Write the *swap* method.

[8 marks]

- (b) I intend to write a *fact* method that will take a positive integer *a* and will return an integer calculated by multiplying *a* by itself and by all positive integers smaller than itself, so that *fact*(4) will return 24, ie 4x3x2x1. If *a* = 0 the method will return 1. While the method requires that *a* is greater than or equal to zero, it does not check to see if the input is correct.

Write the signature of the method and document it using **REQUIRES**, **EFFECTS** and **MODIFIES** comments.

[8 marks]

- (c) Write the *fact* 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) In a 400 x 400 JFrame, display a rectangle with the parameters 20,40,20,30 [10 marks]
- (b) Place a button with no functionality into the JFrame; [5 marks]
- (c) Add the following functionality to the button: when it is pressed the parameters of the rectangle change to 20,40,200,300 [10 marks]

## SECTION B

### Question 6

a)

i) What is an *infix* function? Give one example of a predefined infix function in SML.

[3]

ii) Describe, giving examples, the concepts of *association* and *precedence* that need to be considered when defining a function as *infix*.

[4]

iii) Describe the syntax required to convert a function of two variables into an *infix* one, explaining clearly how issues of operator precedence are dealt with.

[4]

iv) Define an infix function  $p$  that given two integers, ( $x$  and  $y$ , say) returns the difference between their cubes so that  $1 p 2$  results in  $-7$  (that is  $1-8$ ).

[2]

v) Explain how two different results might be expected of the evaluation of  $1 p 2$   $p 3$  and the circumstances under which they might be obtained.

[2]

b)

i) Explain the concept of currying a function, giving examples and explaining the notation used in SML.

[6]

ii) Consider the SML functions defined by:

`fun h(x, y) = x*x*x - y*y*y;`

`fun g x y = x*x*x - y*y*y;`

give, with explanations, the results (values, types and errors returned) of evaluating the following:

`h;`  
`g;`  
`h 4;`  
`g 4;`  
`h 2 3;`  
`g 2 3;`  
`h(3, 4);`  
`g(3, 4);`

[4]

Question 7

a) SML is said to be a *strongly typed language*. Giving suitable examples, explain what this means and give an example of a feature of SML that breaks the ideal of strong typing.

[4]

b) Consider the SML code:

```
datatype coin = nocoin | penny | nickel | dime | quarter | half_dollar | dollar;  
  
abstype purse = broke | add of coin * purse  
with  
val broke1 = broke;  
fun member(x, broke) = false |  
    member(x, add(element, purse)) = x = element orelse member(x, purse);  
fun empty(broke) = true | empty(purse) = false;  
fun next(add(element, purse)) = element;  
fun add1(element, purse) = add(element, purse);  
end;
```

i) Explain the meaning and use of each line of the SML code above.

[5]

ii) Give the output produced by the following evaluations:

```
broke1;  
val a1 = add1(dime, broke1);  
val a1 = add1(penny, a1);  
val a1 = add1(nickel, a1);  
val a1 = add1(dollar, a1);  
if not (empty(a1)) then (a1, next(a1)) else (broke1, nocoin);
```

[5]

iii) In the USA coins are issued in denominations of 1 cent, 5 cents, 10 cents, 25 cents and 50 cent known as penny, nickel, dime, quarter and half-dollar respectively. Write an SML function *value(purse)* which, given a *purse* returns the total value of its contents in cents. For example *value(add1(penny, add1(dollar, add1(nickel, empty1))))*; should return 106. You may need to write helper functions to do this.

[11]

Question 8

a) Distinguish between the SML concepts of List, Tuple and Record, giving examples of their definition and use.

[10]

b) Define a record type 'menu' that holds the following pieces of information: *name, cost, time taken to prepare, and ingredients (with their amounts)*. You will need to decide on the possible structure of the last of these.

[6]

c) Define functions that extract each component.

[2]

d) Define a function *costofingredients* that, given such a record and a list containing ingredients and their unit costs, produces the cost of the ingredients required for the menu. Give an explanation of how your function works.

[7]

Question 9

a)

- i) Give the rules that determine when two Prolog terms match.

[4]

- ii) State which, if any, of the following pairs match, giving your reasoning fully for each pair:

$h(i,j(M))$  with  $h(K, L)$ .

$m([n|p],q)$  with  $m([P|R],S)$ .

$t(u(V))$  with  $t(U(v))$ .

$z(b,[c, D],k(F,g))$  with  $z(b,D,k(f,C))$

[6]

- b) Consider the following Prolog predicate:

```
equal(X,X).  
p(H, [], [H]).  
p(H, L, [L|H]).  
p(H, [T | F], [R|F]) :- p(H, T, R).
```

- i) Explain why the query `equal([a|1], X)` results in `X = [a | 1]`.

[1]

- ii) Describe the relationship between the three parameters of `p` above.

[6]

- iii) Trace the execution of the following query if ';' is typed after each solution is presented: `p(L, [a,b,c], L)`.

[4]

- iv) Write a Prolog predicate `cycle(L, R)` which takes two lists as parameters and returns list `R` that is the same as `L` but with `L`'s first element moved to its end so that, for example `cycle([a,b,c,d], R)` would result in `R=[b,c,d,a]`. [Hint: you may have to define a predicate that appends two lists]

[4]

Question 10

According to p79 of the guide, the lack of strong typing in Prolog has the following consequences:

- “1) the language is simpler – there is less to learn
- 2) the language is more flexible .....
- 3) the interpreter notes fewer mistakes”

- a) Explain the phrase ‘lack of strong typing’, and give an example of this. [5]
- b) Explain, giving examples where possible, the three consequences given above and their advantages and disadvantages. [15]
- c) Distinguish between the declarative and the procedural readings of a Prolog program, and explain why a programmer needs to be able to use both readings when writing or debugging Prolog programs. [5]

**END OF EXAMINATION**