UNIVERSITY OF EDINBURGH course CS0077

FACULTY OF SCIENCE AND ENGINEERING

DIVISION OF INFORMATICS

Computer Science 1 Ah

Degree Examination
Specimen Solutions

Date: Saturday 25th May 2002

Time: 09:30–11:00 (one and a half hours)

Place: Adam House

Room: Ground Floor

Notes about Specimen Solutions

1. These specimen solutions are issued for *guidance only*, to help as a revision aid. They do not represent a complete picture of how the exam was marked; they are specimen solutions, not a marking guide.
2. The solutions typically present one answer from a set of possible answers. Often, answers which have alternative wording or different technical details are equally acceptable.
3. If you have any questions concerning these solutions, please contact the **course organiser**. Please note that the course organiser will not be able to answer questions about the marking of the exam.

Question 1

(a) For each of the following kinds of data, give the *most appropriate* type or class for representing values in Java. Take care with capitalization; remember that Java identifiers are case-sensitive.

i. A sequence of characters representing a line of text that was filled in on a web page form.

`String` [*1 mark*]

ii. The CO_2 emissions of a car, measured in kg/km, accurate to three decimal places (typical emissions figures range from 120g/km – 250g/km).

`float` [*1 mark*]

iii. The sequence of airports visited using an *unlimited* around-the-world air ticket. Your answer should work for any choice of class used to represent an airport.

`java.util.LinkedList` [*1 mark*]

iv. The representation of a 100×100 pixel *greyscale image*, where a grey pixel is represented as an 8-bit number.

`byte`[100][100] [*1 mark*]

v. The representation of a 100×100 pixel *image mask*, which defines a portion of an image (each pixel may be in the mask or not).

`boolean`[100][100] [*1 mark*]

vi. The representation of a 100×100 pixel *colour image*, using a class `Colour` whose objects represent colours.

`Colour`[100][100] [*1 mark*]

(b) In programming, we often examine a value of a type in a particular way, according to the type of the value. In Java, particular control structures are associated with different types of value and the way we might examine them. Complete the statements below which illustrate this association, showing appropriate control structures for processing values of the given type. [*4 marks*]

I would use a ***`for`**-loop* to process a value of type `int[]`.

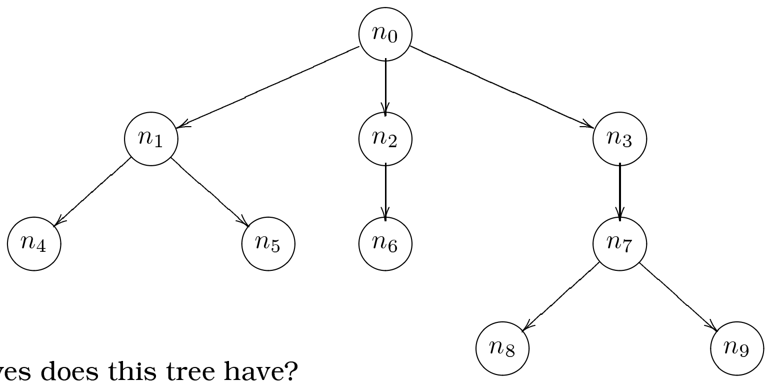
i. I would use a ***`if`**-statement* to process a value of type `boolean`.

ii. I would use a ***`while`**-loop* to process a value of type `StreamTokenizer`.

iii. I would use a ***`while`**-loop* with an ***`Iterator`** object* to process a value of type `Vector`.

Question 2

(a) Consider the tree shown below.



i. How many leaves does this tree have?

5

[*1 mark*]

ii. If the nodes of the tree were rearranged to make a maximally-tall binary tree, how many leaves would the rearranged tree have?

1

[*1 mark*]

iii. Assuming there is an ordering on subtrees from left to right, show the order that the nodes would be visited in a *pre-order* traversal. [*2 marks*]

n_0, n_1, n_4, n_5, n_2, n_6, n_3, n_7, n_8, n_9.

(b) Complete the following method for counting the number of nodes in a binary tree (`BinTree` is a class with two `BinTree` fields `left` and `right`).

```
  private static int size(BinTree tree) {
    if (tree != null ) {
```

[*1 mark*]

```
      return 1 + size( tree.left ) + size( tree.right );
```

[*2 marks*]

```
    } else {
        return 0;
```

[*1 mark*]

```
    }
  }
```

(c) Give the definition for an instance method `size` in the `BinTree` class, which uses the above static `size` method to calculate the size of a non-empty tree object. [*2 marks*]

```
public int size() {
  return size(this);
}
```

Question 3

(a) Place a tick in the box to indicate whether the following statements about Java exceptions are true or false. [*5 marks*]

	true	false
`Exception` is a *class* of the Java language.	✓	
It is possible to invoke methods on exceptions.	✓	
It is possible to *exit* a loop in the middle by throwing an exception.	✓	
It is possible to *enter* a loop in the middle by throwing an exception.		✓
All exceptions a method can throw must be specified with **`throws`**.		✓

(b) Complete the following template to produce a Java program which reads all of the `Student` objects from an object input stream and then closes the stream at the end.

```
import java.io.*;
class StudentReport {
   public static void main(String[] args) {
      try {
         /* Read the file of student records */
         String name = "Students.dat";
         FileInputStream file =
              new FileInputStream(name)  ;                    [1 mark]
         ObjectInputStream in =
              new ObjectInputStream(file)  ;                  [1 mark]
         while (true) {
            try {
               Student s =  (Student) in.readObject()  ;
                                                              [1 mark]
               System.out.println(s);
            } catch (EOFException e) {
               System.out.println("End of report");
               break;                                         [1 mark]
            }
         }
         in.close();                                          [1 mark]
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

Question 4

(a) What operations take place when a Java method with parameters is invoked? [*3 marks*]

Firstly, the actual parameter expressions are evaluated in left-to-right order to give the values which are to be passed to the method. The formal parameters are given these initial values and the method body is executed. If the formal parameter variables are updated in the method body this change is not reflected in the actual parameters which were used for this invocation.

(b) Consider the following Java program. What output does it produce? [*2 marks*]

```
class Hello {
    static void copy (String s) {
        s = s + s;
    }
    public static void main (String[] args) {
        String hello = "Hello";
        System.out.println(hello);
        copy(hello); copy(hello);
        System.out.println(hello);
    }
}
```

```
Hello    // Java calls by value so the assignment to the String parameter s of
Hello    // the copy method does not update the value of hello in main
```

(c) What does it mean for a Java method to be marked as **`static`**? [*3 marks*]

A static method belongs to a class, not to objects of that class. A static method can use other methods, fields and constructors; but only if they are also static.

(d) Consider the following Java program fragment.

```
public class Outer {
  public static class Inner {
    public static int four() {
      return 4;
    }
  }
}
```

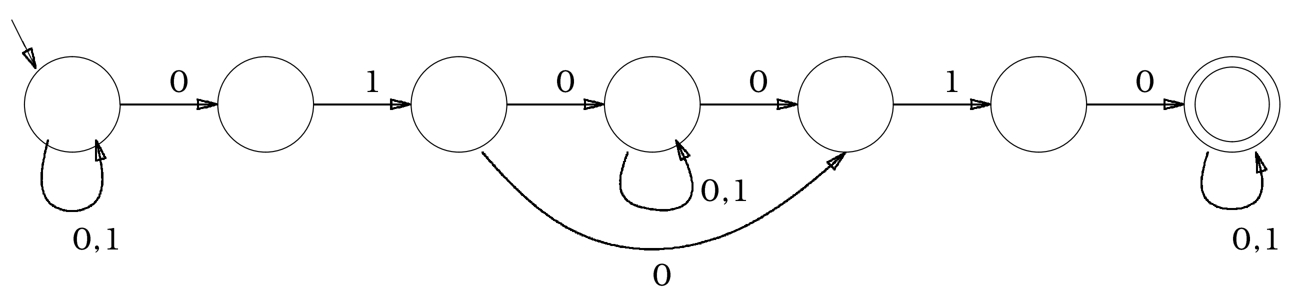
How would the method `four()` above be invoked from another Java class? [*2 marks*]

`Outer.Inner.four()`

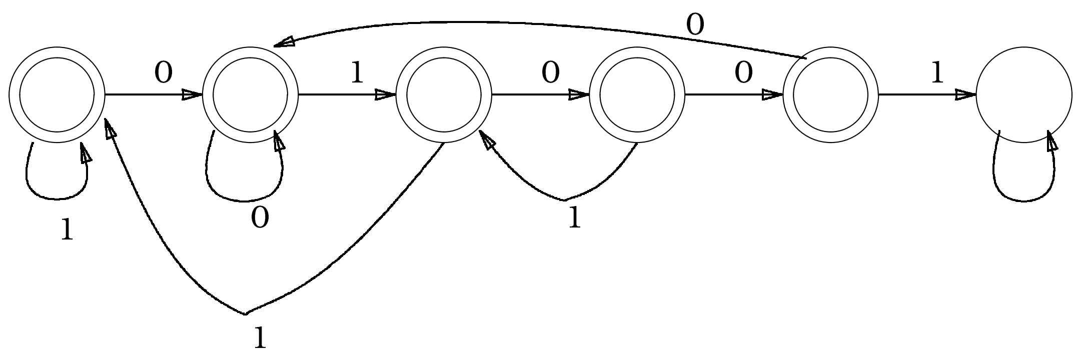
Question 5

(a) Construct finite state machines with the input alphabet $\{0,1\}$ that:

i. Accept exactly those strings that include two occurrences of the pattern `010`. Note that the occurrences could be overlapping. [*3 marks*]



ii. Accept those strings *not* containing an occurrence of the string `01001`. [*2 marks*]



(b) Can a finite state machine accept exactly the following sets of strings? Place a tick in the box if so and a cross otherwise.

i. all strings s of `0`s and `1`s where in any prefix r of s the difference between the number of `0`s and the number of `1`s lies between -3 and +3.. [*1 mark*]

✓

ii. all strings of `0`s and `1`s where the difference between the total number of `0`s and the number of `1`s lies between -3 and +3. [*1 mark*]

✗

(c) Write a regular expression that describes all strings containing just `0`s and `1`s where the number of `0`s is odd or the number of `1`s is even. [*3 marks*]

$1^*01^*(01^*0)^*1^* + 0^*(10^*1)^*0^*$

Question 6

Consider the following brief description of the accident and emergency service at a hospital:

> *Patients arrive at the emergency service. On their arrival they are registered as a case to be dealt with by the service, they are first assessed (a process called triage), their details are taken (name, address, doctor) and allocated to a particular group. The triage groups are: immediate, urgent and non-urgent. Each patient is then treated in the emergency unit and after treatment the case is closed. On the closing of the case the time of closing is compared with the time the case was created to help in the calculation of waiting time statistics for different triage groups.*

(a) Use the techniques of identifying *noun phrases* to identify the classes one would need to provide a model of this system. List the names of the classes you think you would require. [*5 marks*]

I'd expect the students to identify at least: Patient, Case, Triage Group, Time, Details. In marking, allocate 3 marks to getting approximately this list and a further 2 marks to excluding irrelevant noun phrases.

(b) Show how you would implement these classes in Java. Give the class definitions with the main attributes for each class. You do not need to identify any methods in the classes you define. [*5 marks*]

I don't expect perfect Java in the answers but I expect to see structure in the instance variables. In particular that Case should have instance variables for the patient, triage group and start and finish times. In marking I'd allocate 2 marks for getting the Java roughly right and having some instance variables and the additional 3 marks to getting the more complex substructure correct.

Question 7

(a) Explain informally how the *selection sort* algorithm for sorting the elements of an array works.
[*3 marks*]

Search array for position of smallest element, then swap its contents with the first element. Repeat process for the remaining portions of the array until array is sorted.

(b) What is the worst case run time (as a 'big-oh' expression in terms of n) for sorting n elements with selection sort?
[*1 mark*]

$O(n^2)$.

(c) Show how selection sort sorts the following array by writing down the contents of the array each time an element is moved.

4	15	1	16	10

[*6 marks*]

4	*15*	*1*	*16*	*10*
1	*15*	*4*	*16*	*10*
1	*4*	*15*	*16*	*10*
1	*4*	*10*	*16*	*15*
1	*4*	*10*	*15*	*16*

Question 8

A simple hot drink vending machine supplies either tea or coffee when a user inserts money, then presses an appropriate button. For simplicity, the machine only accepts one type of coin. It can supply tea if at least one coin has been inserted and can supply coffee if at least two coins have been inserted. No more than three coins worth of credit can be outstanding at any time. The input actions of the machine are:

coin The user inserts a coin.

teaAsk The user presses the tea button and expects to get a cup of tea if enough money has been inserted.

coffeeAsk The user presses the coffee button and expects to get a cup of coffee if enough money has been inserted.

refund The user presses the refund button and expects all unspent inserted money to be returned.

The output actions are:

teaSupply Supply a cup of tea to the user.

coffeeSupply Supply a cup of coffee to the user.

return Return all unspent money inserted.

(a) Draw a finite state machine for this system. Be careful to take account of all possible sequences of input actions. [*7 marks*]

Four states, $\{C_0, C_1, C_2, C_3\}$, corresponding to zero, one, two or three coins worth of credit. Transitions on input actions are straightforward, given this.

(b) Write down a sequence of input actions that causes every state in your machine to be visited. [*3 marks*]

coin, coin, coin, refund.

Question 9

(a)

i. Draw a truth table for the logical expression (¬A ∨B) ⇔(B ⇒A), including columns for the intermediate sub-expressions ¬A, ¬A ∨ B, and B ⇒ A.

A	B	$\neg A$	$\neg A \vee B$	$B \Rightarrow A$	$(\neg A \vee B) \Leftrightarrow (B \Rightarrow A)$
f	f	t	t	t	t
f	t	t	t	f	f
t	f	f	f	t	f
t	t	f	t	t	t

Award 2 marks if minor errors (e.g. truth table of ⇒ incorrect, 1 mark if serious problems.

[*3 marks*]

ii. Is it a tautology? Give a one line explanation of your answer.

No, because is not true for all truth assignments of A and B.

[*1 mark*]

(b) Using the algebraic laws of boolean logic, simplify the logical expression $A \vee (\neg A \wedge B)$, writing each step of the simplification on a separate line.

$$\begin{aligned} A \vee (\neg A \wedge B) &\Leftrightarrow (A \vee \neg A) \wedge (A \vee B) \\ &\Leftrightarrow t \vee (A \vee B) \\ &\Leftrightarrow A \vee B \end{aligned}$$

[*2 marks*]

(c) Assume variable `i` has initial value `0`. What is the value of `i` after evaluating each of the following boolean expressions in Java?

i. `i++ == 0 & i++ == 1` — *2*

ii. `i++ == 1 && i++ == 0` — *1*

iii. `i++ == 0 && i++ == 1` — *2*

iv. `i++ == 1 || i++ == 0` — *2*

[*4 marks*]

Question 10

(a) For each of the following recursive methods, enter **Y** in the box if the method `f` terminates if given the value 5 as argument. Otherwise enter **N**.

i.

```
static int f(int i) {
  return f(i - 1) * f(i - 1);
}
```

N

[*1 mark*]

ii.

```
static int f(int i) {
  if (i == 0) {return 1;}
  else {return f(i - 1) * f(i - 1);}
}
```

Y

[*1 mark*]

iii.

```
static int f(int i) {
  if (i == 0) {return 1;}
  else {return f(i - 1) * f(i - 2);}
}
```

N

[*1 mark*]

(b) What is printed out by the recursive method:

```
static void g(int i) {
  System.out.print(i + " ");
  if (i == 1) {return;}
  if (i%2 == 0) {g(i/2);return;}
  else {g(3*i+1);return;}
}
```

when invoked with the value 5?

3 10 5 16 8 4 2 1

[*2 marks*]

[Question continues on next page]

(c) Java defines a `Stack` class including the constructor and methods:

```
  public Stack();
  public void push(Object o);
  public Object pop();
  public boolean empty();
```

Using these, complete the definition of the static method `stackCopy` below. Your `stackCopy` method should create a fresh copy of the `Stack` object argument with elements in the *same* order. Further, it should leave the original stack in its initial state, though it will need to change the original stack during its execution. As necessary, use extra `Stack` objects for temporary storage.

```
public class StackUtils {
  public static Stack stackCopy(Stack s) {
```

```
     Stack aux = new Stack();
     Stack result = new Stack();

     while (!s.empty()) aux.push(s.pop());

     while (!aux.empty()) {
       Object o = aux.pop();
       s.push(o);
       result.push(o);
     }
     return result;
```

Marking scheme:
2 marks *if would compile with at most minor errors,*
1 mark *if returned stack has same elements as initial stack,*
1 mark *if returned stack in original order,*
1 mark *if leaves original untouched.*

```
  }
}
```

[*5 marks*]