Solutions and Marking Scheme for Test

UNIVERSITY OF LONDON

GOLDSMITHS COLLEGE

B. Sc. Examination 21 11 2002

COMPUTING AND INFORMATION SYSTEMS

CIS212(CS52006A)a Progrmming:Advanced Topics and Techniques

Duration: 40-45 minutes

Date and time: 21 11 2002

Answer ALL questions.
Full marks will be awarded for complete answers to THREE questions.
There are 60 marks available on this paper
Electronic calculators may be used. The make and model should be specified on the script and the calculator must not be programmed prior to the examination.
NOTE: Other correct solutions are possible and acceptable.

Question 1 (a) We normally mean that the number of comparisons required by the searching algorithm is relatively small. For example, a searching algorithm requires an O(log n) number of comparisons is *more efficient* than the one that requires an O(n) number of comparisons, where n is the number of items (data) in the searching space. [5]

(b) Algorithms 2 is more efficient. [3/10]

The reasons: [7/10]

Algorithms 1 requires $3 + 2N + 1$ execution steps while Algorithms 2 only requires 2 execution steps.

(c) Comments: [3/10]

This is an example of an *inefficient* recursive algorithm. Two main factors contribute to the inefficiency:

overhead The overhead required by every recursive call in the algorithm;

inherent inefficiency the *two* recursive method calls every time in the `else` statement.

The following *iterative* alternative solution would be better in term of efficiency: [7/10]

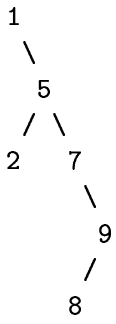
```
public static int iterativeFibonacciTerm(int n) {
// Iterative solution to the Fibonacci problem.
  // initialise base cases:
  int previous=1;
  int current=1;
  int next=1;

  // compute next Fibonacci term when n>=3
  for (int i=3; i<=n; i++) {
    next=current+previous;
    previous=current;
    current=next;
  } // end for
  return next;
} // end iterativeFibonacciTerm
```

Question 2 (a)

```
Red␣+␣Green
```

[5]

(b)



[5]

(c) 2. [5]

If only a difinition is given, then [2/5]

Question 3 (a) An event is an action taken by the user who runs a program. In fact, anything that takes place and that can be 'sensed' by a running program is an event. [5/10]

Examples are such as clicking the mouse on a button, selecting a menu item, pressing a key, or inserting a disk into a drive, etc..

Common events in Java include the followings:

- Button event
- Scrollbar event
- Key pressed event
- Item chosen event.

[5/10]

(b) The method `display`: [10]

```
public static void display(Node L) {
   Node current = L;
   while (current!=null) {
     System.out.println(current.getItem());
     current = current.getNext();
   }  // end while
 } // end display
```