Computer Science 1 Bh

Degree resit examination

Date: Tuesday 19th September 2000
Time: 11:30–1:00 (one and a half hours)
Place: Roxburgh Place
Room: Hall A

Board of examiners
Chair: Stuart Anderson.
External Examiner: Muffy Calder.

Instructions to candidates
1. Check that the question paper contains pages 1–10 after this cover page. If it does not, inform the invigilator.
2. You should attempt as many questions as possible.
3. The questions in this paper vary widely in difficulty. During your first pass through the paper, you are advised not to dwell on questions to which the answer is not readily apparent.
4. Write your answer to each question in the box or table provided. If you wish to write more in answer to a question, continue on the blank page opposite, indicating that you have done so.
5. Please write legible and concise answers.
6. The marks allocated to each part of a question are indicated in the margin. There are 100 marks in total.
Question 1

The java.util package provides the java.util Enumeration interface. This declares two methods: hasMoreElements() and nextElement(). The first of these returns a Boolean value, according to whether the enumeration is exhausted. The second method returns the next element as an Object or throws NoSuchElementException if no more elements exist.

a). Complete the declaration of the class Counter below which implements this interface. If n is a natural number then the use of the constructor method with Counter(n) should return an enumeration that produces, as Integer objects, the first n natural numbers (starting with 0).

```java
import java.util.*;
class Counter implements Enumeration{
    private int count, size;
    public Counter(int size) {
        // provide the implementation of this method
        [2 marks]
    }

    public boolean hasMoreElements() {
        return count < size;
    }

    public Object nextElement() throws NoSuchElementException {
        // provide the implementation of this method
        [4 marks]
    }
}
```

b). Write a Java implementation of a method Sum() which takes an enumeration that, like Counter(n), enumerates a collection of Integer objects, and returns their sum, as an int.

```java
```
Question 2

This question concerns stack-based evaluation of arithmetic expressions.

a). Draw syntax trees for the following expressions:
   - \((2 \times 4) + (5 \times 3)\)
   - \(2 \times (4 + 6) \times 3\)

b). Give the expressions derived from each of these trees by i) infix and ii) postfix tree traversal.

c). Describe the instructions and operation of a simple abstract stack-machine suitable for evaluating such expressions. Using the expression \((2 \times 4) + (5 \times 3)\) as an example explain how to generate instructions for your machine.
Question 3

The skeleton method given below is intended to determine how to pack a number of objects whose sizes are given in the vector size into a space of size n. The aim of the method is to pack the space until no more objects can fit into the remaining space leaving as much wasted space as possible. This is achieved by packing some (but not necessarily all) of the objects whose sizes are given from element i and upwards of the vector size. The method returns the amount of space left over as its result. This approach will tend to favour filling up the space with lots of small items to “squeeze out” some big items. This might be an appropriate strategy where storing small items is of greater benefit than large items. The method is intended to use recursion to calculate the answer. Your task is to fill out the boxes in the skeleton with code which defines the method correctly.

```java
public int badPack(Vector size, int i, int n) {
    // Returns the space left when we try to pack objects whose sizes are
    // given from element i of size upwards. The size of the space we are
    // packing into is n.
    if (n == 0) {
        // We have filled up the container exactly.
    }
    else if (i == size.size()) {
        // We have run out of things to pack
    }
    else {
        // Compute the size of the next element
        int sizeOfI = ((Integer)size.elementAt(i)).intValue();
        if (sizeOfI > n) {
            // The next object is too big to fit in the remaining space
        }
        else {
            // We can choose either to include the ith object or not.
        }
    }
}
```
Question 4

a). The method given below is intended to find the index of the first occurrence of the minimum value in a non-empty vector of Integer objects.

```java
public int minIndex(Vector v) {
    int result = 0;
    int minv = ((Integer)v.elementAt(result)).intValue();
    for (int i = 1; i < v.size(); i++) {
        int val = ((Integer)v.elementAt(i)).intValue();
        if (val <= minv) {
            result = i;
            minv = val;
        }
    }
    return result;
}
```

i). Describe a sequence of integers that will give the correct result using the above program.

ii). Describe a sequence of integers that will give incorrect results using the above program.

iii). Outline how you would go about repairing the fault in the program so it would give correct results with both your tests.

b). Describe the four main activities involved in debugging.
Question 5

a). Describe the roles of the clients and the server in a client/server environment. [2 marks]

b). Describe the information which is needed in order to make a socket connection from one machine to another. [2 marks]

c). Assuming that a socket connection has been successfully established between a client and a server, what problem will occur when the following two code fragments are executed?

Server side:

```java
String line = input.readLine();
client.println ("Message received");
client.flush ();
```

Client side:

```java
String line = input.readLine();
server.println ("Communication established");
server.flush ();
```

[4 marks]

d). How are communications problems such as the one which you identified in part c) resolved in practice? [2 marks]
Question 6

a). Supply a Java method which would produce the following Java byte code method when compiled.  
**Note:** if \textit{ge} compares the top of the stack with zero; \textit{iconst.m1} pushes the integer value –1 onto the stack; and \textit{imul} multiplies the two integers on top of the stack.  

```
Method int abs(int)
0 iload_1
1 if.ge 8
4 iload_1
5 iconst.m1
6 imul
7 istore_1
8 iload_1
9 ireturn
```

b). Supply the compiled Java byte code equivalent of the following Java method.  

```
int maxZero (int x, int y) {
    if (x > y)
        return 0;
    else
        return y;
}
```
Question 7

a). Consider a Java virtual machine which has the following values in memory and on the stack.

<table>
<thead>
<tr>
<th>memory</th>
<th>stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2</td>
<td>10 2</td>
</tr>
</tbody>
</table>

Giving this starting state, show the intermediate states of the memory and the stack as each instruction is executed. [5 marks]

1. istore_2
2. iconst_1
3. iload_1
4. iinc 1 1
5. istore_2
6. iload_1
7. istore_2
8. iconst_3

b). List two advantages of compilation to byte code over compilation to native code. [2 marks]

c). List one disadvantage of compilation to byte code over compilation to native code. [1 mark]

d). What is the role of a just-in-time compiler in a Java virtual machine? [2 marks]
Question 8

a). In informal terms, what is Java’s notion of a graphics context? [2 marks]

b). What is the class of a graphics context object? [1 mark]

c). You have been given the task of writing a simple graphics application which prints random squares of random colours at random positions in a 500 × 500 grid. You are given the methods `rnd()` and `rndC()` as shown below.

```java
// A method to return random numbers between 0 and 500
int rnd() {
    return (int) (500 * Math.random());
}

// A method to return random numbers between 0 and 255
int rndC() {
    return (int) (255 * Math.random());
}
```

Provide a `paint()` method which repeatedly performs this task. You will need to use Java’s `drawRect()` method with specification `void drawRect (int x, int y, int width, int height)` and also its `setColor()` method with specification `void setColor (Color c)`. Note carefully that you are to draw squares and not arbitrary rectangles. [7 marks]
Question 9

You are given the recursive method

```java
public int f(int n) {
    if (n==0) return 0;
    if (n==1) return 1;
    if (n==2) return 2;
    else return f(n-1) + f(n-2) + f(n-3);
}
```

a). Draw a tree representing the recursive calls to `f()` when evaluating `f(5)`. [4 marks]

b). Why is the method `f()` inefficient? [2 marks]

c). Using dynamic programming or otherwise, write a more efficient version of `f()`. [4 marks]
Question 10

a). You are given a box containing many processors which you suspect may not have been checked for errors by the manufacturer. If the probability of an error within a processor is one in 10, how many chips do you have to test to be satisfied that there is less than one in a thousand chance that the box has not been checked? [5 marks]

b). Explain why independence is important in Monte-Carlo algorithms. [3 marks]

c). Why are random numbers hard to generate? [2 marks]