Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Software Evolution

Date: Friday 30th May 2008
Time: 14:00 – 16:00

Please answer any THREE Questions from the FOUR questions provided

This is a CLOSED book examination

The use of electronic calculators is NOT permitted
1. a) Draw a data dependency graph for the following fragment of code. (14 marks)

```java
public int colWidth(MyTable table, int colNum, int margin) {
    Column col = table.getColumn(colNum);
    int width = 0;
    Renderer renderer = col.getHdrRenderer();
    if (renderer == null) {
        renderer = table.getDefaultRenderer();
    }
    Component comp = renderer.getCellComp(0);
    width = comp.getBestSize();
    for (int r=0; r<table.getRowCount(); r++) {
        renderer = table.getCellRenderer(r, colNum);
        comp = renderer.getCellComp(colNum);
        width = Math.max(width, comp.getBestSize());
    }
    width += 2*margin;
    return width;
}
```

b) The data dependency graph you have drawn for part a) indicates an opportunity for optimising the code in a small way. Describe this opportunity, and the features of the data dependency graph that suggest it. (3 marks)

c) If translated naively from the source code, compiled object code can spend a lot of time in allocating and deallocating memory space for variables that may only be used for a short time. Therefore, an optimising compiler will look for opportunities to reuse memory where possible: for example, where two variables of the same type are declared but never co-exist in memory. By themselves, data dependency graphs are not sufficient to identify such optimisation opportunities. Explain why this is the case, and suggest another kind of graph that can be combined with data dependencies in order to provide the information necessary for this kind of optimisation. (3 marks)
2. The following fragment of code is taken from an open source implementation of a Sudoku puzzle solver. For those not familiar with Sudoku, the game is played on a grid of cells, divided into 9 3x3 sub-grids. At the start of the puzzle, a few of the cells will have been given values (digits from 1 to 9) while others will have been left blank. The aim of the puzzle is to fill the blank cells with single digits so that every row, every column and every 3x3 sub-grid contains exactly one occurrence of each of the digits from 1-9. An example Sudoku puzzle (taken from Sudoku.com) is shown after the code listing, for illustration.

```java
public class SudokuSolver {
    ...

    private static void solve(Grid grid, List<Grid> solutions) {
        if (solutions.size() >= 2) {
            return;
        }

        XXXXX loc = grid.findEmptyCell();

        if (grid.invalid(loc)) {
            solutions.add(grid.clone());
            return;
        }

        for (int n=1; n<10; n++) {
            if (grid.set(loc, n)) {
                solve(grid, solutions);
                grid.clear(loc);
            }
        }
    }
    ...
}
```

<table>
<thead>
<tr>
<th>6</th>
<th>1</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

(Question 2 continues on the following page)
(Question 2 continues from the previous page)

(a) Point out (by giving line numbers) an example of one of the standard idioms covered in the lecture course within the code fragment. Give a language-independent statement of the idiom (e.g. in pseudocode) and state which variables and expressions in the code fragment correspond to each component of the idiom. (5 marks)

(b) The details of the data structure used to represent the location of a cell in the grid have been hidden in the program listing (with xxxx). Hypothesise two substantially different ways in which the location might be represented by the rest of this program, based on your understanding of the Sudoku puzzle and the clues given by the fragment of code shown. Your hypotheses should include a suggestion for an invalid location, as tested for on line 23. For each hypothesis, describe the features you would look for in the remainder of the code to prove or disprove your hypothesis. (8 marks)

(c) Somewhat surprisingly, this code fragment contains no explicit checks that the rules for digit placement hold. Based on the clues present in the code, hypothesise where in the remainder of the Java program this behaviour is implemented. Justify your answer. (3 marks)

(d) This code fragment actually contains two examples of a standard idiom: one in the form we encountered during the lectures and one in recursive form. Reformulate the standard language-independent pseudocode statement for this idiom into recursive form, based on the code pattern suggested by this example. In your reformulation, it should be possible to clearly identify the main components of the idiom. (4 marks)

3. Wolf, Inc., is a major international provider of online and broadcast content. Their business model involves the development and production of high-profile TV series, which can be sold around the world, and which are supported by a rich array of digital content, ranging from fan sites and review sites through to tie-in games, ring-tones, wallpapers and other forms of digital merchandising.

Until recently, the two sides of the business (TV series production and associated digital content) were kept separate, and as such were supported by independent computer systems. Now, however, the situation has changed. Writers have demanded and won the right to be paid royalties when their work is shown through online channels, as well as through the more conventional broadcast mechanisms. This requires a more joined-up computer system than Wolf, Inc., currently operates.

(Question 3 continues on the following page)
After a couple of abortive attempts to modify the existing systems to provide the required functionality, it has been decided to re-engineer the system using the Chicken Little methodology. In addition to providing unified royalties processing, Wolf also wishes to take advantage of this re-engineering effort to add functionality to manage productions more carefully throughout their whole lifetime—in particular, setting deadlines for writing contracts and monitoring how well these are met. Wolf would then be able to use a writer’s track record on delivering quality material by the deadline in order to judge who should be given contracts for new work.

You have been put in charge of the team responsible for planning and undertaking this re-engineering project. Your task is to bring the new functionality on-line by the deadline for introduction of the new royalty payments (in 6 months’ time), while allowing the current business of the company to proceed unimpaired. As a first step, you ask your team to evaluate the architecture of the current computer systems. The result is shown below.

As the diagram shows, the system currently accesses two different data stores, each of which is stored using a different persistent technology. Each type of user (including Writers and Agents, who are not Wolf employees, but who have access to their computer system in order to place bids for writing jobs) has their own system with strictly limited functionality. Writers and Agents cannot get information about their royalty payments, for example, and there is no realistic way to get an overview of the behaviour of the whole enterprise for use by senior management. The royalties system also makes use of a feed from royalty watch, the industry standard body that keeps track of broadcasts of material by different organisations after purchase. This provides the raw data for the calculation of royalty payments.
(Question 3 continues from the previous page)

a) How would you characterise this architecture in terms of its decomposability? (2 marks)

b) Amongst other changes, the system will be re-engineered to operate over a single database. This will allow the development of new functionality to support senior staff in decision making. In addition, new types of data must be stored in the database, in order to help the company track non-conventional broadcasts/uses of a writer’s work, so that the correct royalties can be paid. The third piece of new functionality is a system to manage contracts throughout their lifetime, tracking deadlines and late pieces of work, etc. This will also require the storage of new data in the database, plus substantial new code to be written.

Based on these considerations, discuss the suitability of each of the three migration strategies (i.e. forward migration, reverse migration and general migration) for this re-engineering project. (8 marks)

c) After further work with your team, you come up with the following target architecture for the new system:

(Question 3 continues on the following page)
Using your preferred migration strategy from part b) of the question, design a migration path for this system that brings the important functionality on-line soonest, while maintaining existing mission-critical functionality as necessary. For each migration step, you should:

- Draw a diagram of the architecture of the system at that point in time, showing clearly any gateways in place, and distinguishing legacy from target components and reverse from forward gateways.

- justify your choice of component to migrate in this step in terms of the benefits, costs and risks to the business. (10 marks)

4. The tourism council of a major European capital wishes to make use of the range of information available within its information systems to provide web-sites that encourage tourism within the city. As a first attempt, the company wishes to create a web site (called DaysOutWeb) that integrates data from BusDB, a database containing information about bus routes and fares within the city, with data from DestinationsDB, a separate database containing information about various tourism destinations throughout the city, plus their opening hours and cost of entry. DaysOutWeb would use this information to help visitors to the city to plan day trips by listing tourist sites that are easily reachable from their accommodation, and giving a cost for the full trip and the numbers of the buses that should be used to get there.

Since both BusDB and DestinationDB are well-used systems with little spare capacity to support a new set of users, the team charged with implementing DaysOutWeb has decided to base their application on a new database, populated by regular feeds from the two main databases. The initial architecture design for the system is as follows:

![Architecture Diagram]

Note that two separate feeds are needed to provide the raw data for DaysOutWeb, one from each of the main data sources.

(Question 4 continues on the following page)
(Question 4 continues from previous page)

a) As can be seen from the initial data design, the team initially did not consider the need for any data cleaning elements within the ETL feed processes. Name three kinds of data quality problem that might be encountered by the users of DaysOutWeb because of this omission. For each problem kind, give an example of a concrete data quality issue that might occur in this domain, and describe its possible effects on the users of DaysOutWeb. (6 marks)

b) For each of the following scenarios, determine the type of feed that would be needed at each of the two feed points (i.e. push, pull or hybrid). Describe a possible feed strategy for each source database, explaining when the data migration would take place, and what guarantees you could give to the users of DaysOutWeb about the freshness of the data it contains.

i) In this scenario, BusDB is an aging COBOL-based database system which is updated every night, between 3.00am and 5.00am. Each such update will typically affect no more than 100 records. No trigger mechanism is supported for BusDB, but a log of the changes made during the nightly update is available for access for 3 hours from the end of the update period, before being archived to tape.

DestinationDB, on the other hand, is a relational database, with trigger capabilities. Updates occur infrequently (an average of only 10 records affected per week), but on no regular schedule, as they are made manually by data entry staff. The load on DestinationDB is relatively light throughout the day, and almost non-existent outside working hours.

ii) In this scenario, BusDB has the same properties as in scenario i).

DestinationDB, on the other hand, is a very heavily used system in this scenario, and the DB administrator has refused to allow triggers to be enabled, because of the negative effect on the performance of the system as a whole. Instead, a weekly change log is made available during a 2 hour window, every Wednesday from 12.00pm to 2.00am, for retrieval by feeds such as DaysOutWeb. Updates to DestinationDB occur at unpredictable intervals in this scenario, but are much more frequent, averaging changes to 250 records per day. (8 marks)
c) After initial trials of the feed architecture pictured above, it soon becomes apparent that the quality of data in the two sources is too poor to be used directly in DaysOutWeb. A data cleaning component must be added to the feed process. Redraw the architecture for the feed, showing the placement of the data cleaning code relative to the other feed components. Give a brief description of how the cleaning component would handle one of the data quality problems you identified in your answer to part a) of this question (e.g. when is it triggered, what inputs does it take and what happens to its outputs). (3 marks)

d) Outline a strategy that could be employed by the data cleaning component to ensure that the destinations that are stored in the DaysOutWeb database correspond to real locations. (3 marks)