



UNIVERSITY *of* LIMERICK

O L L S C O I L L U I M N I G H

Faculty of Science and Engineering

Department of Computer Science
and Information Systems

Repeat Examination Paper

| | | | |
|-------------------|--|-------------------------|---------------|
| Academic Year: | 2007/08 | Semester: | Summer |
| Module Title: | Software Testing and Inspection | Module Code: | CS4004 |
| Duration of Exam: | 2½ Hours | Percent of Total Marks: | 100 |
| Lecturer(s): | Power & McElligott | Paper marked out of : | 100 |

Instructions to Candidates:

- Answer Q1 and Q2
- State clearly any assumptions you make.

Q1 Functional Testing

(50 marks)

You are involved in testing a system component that accepts an employee's input regarding hours worked in order to calculate the overtime payment for that employee.

Here is an extract from the specification

The system shall prompt the user for the following data:

- Employee number
- Job category
- Week ending (Enter a date of the format: DD/MMM/YYYY)
- Total hours worked
- Weekend hours worked

The following constraints apply:

- Employee numbers are five digits in the range between 23456 and 54765
- There are four different job categories:
 - Grade 1 (50% extra for overtime)
 - Grade 2 (50% extra for overtime)
 - Charge-hand (Charge-hands get 25% extra for overtime)
 - Supervisor (These don't get normal overtime, except 30% for weekends)
- The hours worked cover the whole week, and must not exceed 85
- The weekend hours entered must be included in the Total hours and therefore must be less than or equal to that value.
- Overtime is paid to eligible employees for any hours in excess of 40.

The system shall accept the data from the interface depicted in Figure 1 and use the Employee number to search a file called rates.txt which associates each Employee number with an Hourly rate. Typical hourly rates are 9.50, 10, 11.00, (these do not need to be validated). The system shall use the data entered to calculate the Overtime amount due to each employee for the relevant week).

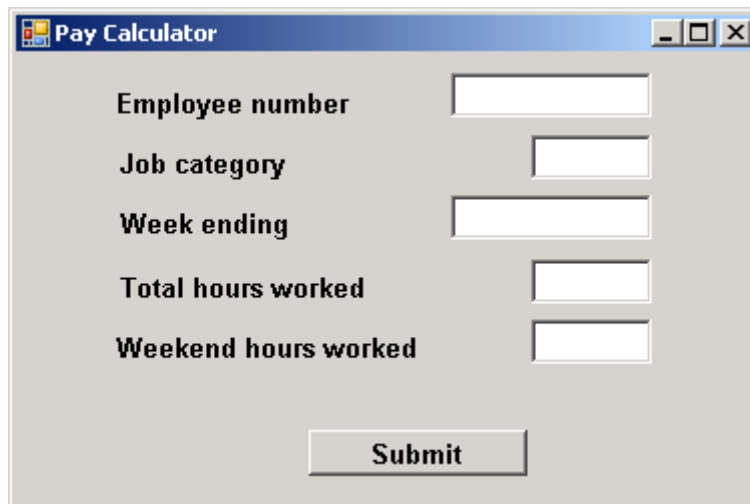


Figure 1

You are required to design test cases for this system using equivalence classes and boundary value analysis. The test cases should be documented as follows:

- for each **equivalence class** you create you should specify its number, its description, whether it is valid/invalid and provide an example. **30 marks**
- a table specifying for each **test case** its number, the test case (i.e., the input values), whether the test case is valid or invalid, the classes covered (including boundaries if any), and the expected outcome. **20 marks**

Q2 Structural Testing**(50 marks)**

- (a) Write test cases to achieve 100% statement coverage (S) of the program shown in Figure 2. For each test case you should write its test case number, its description and expected outcome. Use a table similar to that used in Assignment 2 last semester. **20 marks**
- (b) Draw a Control Flow Graph (CFG) for this program. **10 marks**
- (c) Using your CFG write sufficient test cases to achieve
100% decision coverage (D)
100% condition coverage (C)
100% decision-condition coverage (DC) **20 marks**
- In your answer to this you should indicate whether each particular test case contributes to D, C or DC coverage.

```
1 import javax.swing.JOptionPane;
2 import java.io.*;
3 public class CS4004Exam
4 {
5     public static void main(String [] args) throws IOException
6     {
7         String f1 = "", f2 = "";
8         while (f1.equals(""))
9             f1 = JOptionPane.showInputDialog(null, "Name of 1st file:");
10        while (f2.equals(""))
11            f2 = JOptionPane.showInputDialog(null, "Name of 2nd file:");
12        File file1 = new File(f1);
13        File file2 = new File(f2);
14        String wr;
15        if (file1.exists())
16        {
17            FileReader aFileReader = new FileReader(file1);
18            FileWriter aFileWriter = new FileWriter(file2);
19            BufferedReader aBufferedReader = new BufferedReader(aFileReader);
20            BufferedWriter aBufferWriter = new BufferedWriter(aFileWriter);
21            String wf, output = "";
22            while ((wf = aBufferedReader.readLine()) != null)
23            {
24                wr = "";
25                for (int index = (wf.length() - 1); index >= 0; index--)
26                    wr += wf.charAt(index);
27                if (wf.equalsIgnoreCase(wr))
28                    aBufferWriter.write(wf + "\n");
29            }
30            aBufferWriter.close();
31            aBufferedReader.close();
32            JOptionPane.showMessageDialog(null, "File processed");
33        }
34        else
35            JOptionPane.showMessageDialog(null, "File not found");
36        System.exit(0);
37    }
38 }
```

Figure 2