CARDIFF UNIVERSITY
EXAMINATION PAPER

Academic Year:	2017/2018
Examination Period:	Spring
Examination Paper Number:	CMT304
Examination Paper Title:	Programming Paradigms
Duration:	2 hours

Do not turn this page over until instructed to do so by the Senior Invigilator.

Structure of Examination Paper:

There are 4 pages.
There are 4 questions in total.
There are no appendices.

The maximum mark for the examination paper is 60 and the mark obtainable for a question or part of a question is shown in brackets alongside the question.

Students to be provided with:

The following items of stationery are to be provided:
ONE answer book.

Instructions to Students:

Answer **three** questions.

Important note: if you answer more than the number of questions instructed, then answers will be marked in the order they appear only until the above instruction is met. Extra answers will be ignored. Clearly cancel any answers not intended for marking. Write clearly on the front of the answer book the numbers of the answers to be marked.

Students are permitted to introduce to this examination any textbook, any printed or handwritten notes, and other similar materials. These may be annotated, highlighted and bookmarked as desired.
The use of translation dictionaries between English or Welsh and a foreign language is permitted in this examination.
The use of electronic devices is not permitted.

Q1. (a) A company is to develop an electronic access control system for a server room, thus enabling only authorised personnel to access it at given times. For instance, Bob might be allowed to enter the server room only on Tuesdays.

Is Answer Set Programming a suitable programming paradigm in this case compared to other programming paradigms? [1]

Explain why in at most two sentences. [2]

Write an Answer Set Program to capture the example of Bob above. [10]

(b) What is the stable model of a logic program P? [7]

Q2. (a) (i) Provide a minimal Haskell typeclass definition of a functor and explain what a functor does with an example. [5]

(ii) Provide a minimal Haskell typeclass definition of an applicative and explain what an applicative does with an example. [5]

(b) Consider the following Haskell function reversing the order of the elements in a list:

```
reverse_slow ::  [a] -> [a]
reverse_slow [] = []
reverse_slow (x:xs) = reverse_slow xs ++ [x]
```

(i) What is the time complexity of this function? Justify your answer. [3]

(ii) Write a more efficient version of the function `reverse_slow`. State and justify its time complexity. [7]

Q3. (a) A company needs to develop a *highly efficient* web-based system providing movie recommendations.

Which *programming paradigm* would you recommend they use? Provide *four* justifications for your answer. [5]

(b) What is the main difference between threads and processes in concurrent programming? [5]

(c) What is a critical section in concurrent programming? [2]

(d) What is mutual exclusion in concurrent programming? [3]

(e) In the context of using *formal methods* to develop an *autonomous* robot:

Give *two* concrete examples in each case of how a specification produced by traditional methods can be (a) *incomplete* and (b) *inconsistent*. Also give *one* example of (c) an *unstated assumption*. [5]

Q4. (a) A critical Unix server has experienced a security breach by a local user and you need to know which users were logged into the server at the time. The server logs all logins that happen via SSH in the file /var/log/ssh.log. Usernames and users' full names are stored in the file /etc/passwd. You need to provide a list of full names, usernames, and time stamps, sorted by the user's last name. You cannot install new software on the server.

QUESTION CONTINUES ON NEXT PAGE.

Which programming paradigm or programming paradigms would you use? Provide one justification for each programming paradigm you recommend to use. What are the main building blocks of your solution? Provide at least three building blocks. [5]

(b) You are given the following regular expressions (Perl syntax):

```
\d+(\.\d+)?\s
(\w+)\s+\1
\d+\s*[+-]\s*\d+
```

For each expression, give one explanation in words of what it does and an example string that matches the expression. [3]

(c) Consider the following two problems:
(1) Computing the average of a large set of numerical values.
(2) Finding a computer program that computes an approximation of the average of a set of numerical values.

Which of these problems is more suitable for a genetic algorithm? Give one characteristic of each problem to support your answer. [3]

(d) Give three characteristic building blocks of a genetic algorithm. For each, specify whether or not it should include randomness. [6]

(e) Assume you have a processor with 4 registers, referred to as R1, R2, R3, and R4, and the following instruction set:

`ADD Rx, Ry`: Add the contents of the registers Rx and Ry and store the result in Rx.
`DEC Rx`: Decrease the value in register Rx by 1.
`INC Rx`: Increase the value in register Rx by 1.
`LOAD Rx, Z`: Direct load into register Rx using address Z.
`LOAD Rx, @Ry`: Register indirect load into Rx using register Ry
`LOAD Rx, #Z`: Immediate load of Z into register Rx.
`JMP @LABEL`: Unconditional jump to the address of LABEL.
`JZ @LABEL`: Jump to the address of LABEL if the last operation has resulted in a value of zero.

QUESTION CONTINUES ON NEXT PAGE.

You are given the following program:

```
LOAD R1, #0
LOAD R2, 1
LOAD R3, 2
JZ   @E
@L:
  LOAD R4, @R2
  ADD  R1, R4
  INC  R2
  DEC  R3
  JZ   @E
  JMP  @L
@E:
```

Assume the memory contains the following values (starting from address 1):

```
   4 3 2 1 2 4 5
```

Which function does the program compute in R1? Describe the general function of the program and give the value in R1 after the program terminates for the given memory contents. [3]