CARDIFF UNIVERSITY

EXAMINATION PAPER

Academic Year:	**2016-2017**
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 **five** pages.
There are **four** 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 a translation dictionary between English or Welsh and another language is permitted in this examination.

The use of electronic devices is not permitted.

Q1. (i) Briefly discuss the following points concerning *Scala*:

(a) How can Scala *interoperate* with Java? [4]

(b) Explain four ways in which the *design* of Scala is an improvement over the design of Java. [4]

(ii) What is meant by a *guard* in programming languages? Where can guards be used in *Scala*, and where can they be used in *Haskell*? Give and explain a simple *example* in each language. [5]

(iii) What is meant by *spaghetti code*? [2]

Why is it *bad*? [1]

How do modern languages *prevent* spaghetti code? [1]

(iv) How do the goals of the *javadoc* and *web* documentation tools differ? [2]

In practice, *web* is not often used. What *approach* do programmers often take instead to solving the problem it addresses? [1]

Total Marks [20]

Q2. (i) Using *Lisp*, assuming that all data to be processed are integers,

(a) define a *lambda function* to return the greater of its two arguments [2]

(b) use this lambda function with *folding* to define a function `mymax` which returns the maximum number in a list of numbers; explain how your function works [3]

(ii) Explain very briefly how *call by reference* works. [2]

In an old version of FORTRAN, the following code was written. This version of FORTRAN used *call by reference.* The code was compiled. When it ran, it printed the value 6. Explain carefully how this can be explained in terms of call by reference. (The keyword `CALL` is used to execute the subroutine `WEIRD`). [4]

```
CALL WEIRD(2)
PRINT 2+2
...
SUBROUTINE WEIRD(I)
I = 3
RETURN
END
```

(iii) In Java, the *increment* operator ++ is not an *atomic instruction*. Explain what this *means*, and what the possible *consequences* are. [5]

(iv) Some programs support *plug-ins*, i.e. extra code possibly developed by others, which can work with the program. For example, an image processing program may allow plug-ins which perform various image enhancement tasks. What is the concept of *reflection*, and how is it relevant in this context? [4]

Total Marks [20]

Q3. An *alphametic* puzzle is one in which words are put together into an arithmetic formula; the player must substitute digits for the letters to make the formula true. (Numbers with leading zeros are not allowed). E.g.

```
  SEND
+ MORE
-----
 MONEY
```

has the solution `D=7, E=5, M=1, N=6, O=0, R=8, S=9, Y=2.`

(i) What programming paradigm is most suited to solving this kind of puzzle? [1]

(ii) Write a program to solve the particular puzzle given above in *Prolog*, based on the following ideas:

(a) give Prolog facts to state that each of the numbers 0 to 9 is a *digit*, [2]

(b) define a *rule* to find the answer, which takes into account the ideas that:

each letter is to be replaced by a digit,

the addition of the two numbers represented by `SEND` and `MORE` should give the value represented by `MONEY`,

none of these three numbers should start with zero,

`write` should be used to write out the answer, [6]

(c) *use* that rule. [1]

(iii) In such puzzles it is usual to insist that each letter stands for a *different* digit.

(a) Explain in general terms how to *recursively* process a list to check that each item in a list is different, in terms of head and tail. [4]

(b) Implement this idea as a *Prolog* functor `all_different` which takes a list of items, and is satisfied if each item in the list is different from all other items in the list [4]

(c) Explain how to *modify* your original program to use this rule to ensure each letter represents a different digit. [2]

Total Marks [20]

Q4. (i) A company is to develop an *electronic access control system* for a *bank vault.*

(a) Which *programming paradigm* would you recommend they use? [1]

(b) Give *four* justifications for this choice. [4]

(ii) A company is to develop some software to allow end *users* to *write programs* to efficiently solve *goods delivery* problems. Things to be delivered are in rectangular boxes of various sizes and weights, with a date by which they are to be sent; they are to be loaded into rectangular shipping containers which have various weight limits.

(a) Which *programming paradigm* would you suggest for the language to be used? [1]

(b) Give *three* justifications for this choice. [3]

(c) Explain in outline how such a *program* would be written. [5]

(d) What form would the output of such a program take? [3]

(iii) A company is to develop some software which scans various news feeds, and uses the information to recommend share purchases and sales to the users of the software.

(a) Which *programming paradigm* would you recommend they use? [1]

(b) Give two *key ideas* of this paradigm to justify this choice. [2]

Total Marks [20]