CARDIFF UNIVERSITY

EXAMINATION PAPER

Academic Year:	**2014-2015**
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 **three** 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.

Please turn over

Q1. The following question refers to the *Haskell* programming language.

(i) Suppose we define `productplusone` as

```
productplusone a b = a * b + 1
```

Write down the *type* of `productplusone`, explaining what the type definition means. [7]

(ii) Suppose we now define

```
ppot = productplusone 2
```

(a) What concept allows us to write such a definition? [1]

(b) What does `ppot 5` compute? [1]

(c) What is the type of `ppot`? [1]

(iii) (a) What is the type of the built-in map function? [1]

Suppose that *haskell* did not provide the map function.

(b) Write your own definition for map, using a *list comprehension*, and explain how it works [4]

(c) Write another definition for map, using *recursion* and *pattern matching, and explain how it works.* [5]

Q2. Consider the following *Prolog* definition which is intended for use with the first two arguments instantiated as numbers:

```
mystery(X,Y,Y):-  X =< Y.
mystery(X,Y,X):-  X > Y.
```

(i) Explain what it does by considering what happens if (a) we query mystery with *three numeric arguments*, (b) if we query mystery with the first 2 arguments being *numeric* and the third one a *variable.* [6]

(ii) If this is used in a larger program, it can be *inefficient.* Explain why. [4]

(iii) What Prolog *operator* can be used to overcome this efficiency? Explain how it does this. Write a more *efficient* version of `mystery` using this operator. [4]

(iv) What happens if we call the original version with *all three arguments* as *variables*? Write an improved version that will return false if the *first two arguments* are not instantiated as intended. [6]

Q3. The field of *image processing* considers algorithms to *modify images*, while *computer vision* attempts to *understand* the content of *images*.

(i) Do you consider *filter based computing* to be a good paradigm for designing software for each of these fields? Briefly justify your answers. [6]

(ii) Suppose that you have been asked to develop a program to help home users improve their *snapshots*. Give examples of *four filters* they might find useful. [4]

(iii) A consultancy company designs software systems for *camera based inspection* of manufactured goods, e.g. to check that part of an item is not missing, or that it has been assembled correctly. A bespoke system is provided to each customer. What advantages might development based around a *coarse-grained dataflow* approach offer to the consultancy company? [6]

(iv) Explain why *Quartz Composer* would not directly be a useful tool for the purpose in (iii). How would it need to be *modified* to meet this requirement? Give *two* concrete examples of such modifications. [4]

Q4. In the context of using *formal methods* to develop a *driverless car*:

(i) Explain briefly why *errors* discovered during (a) *testing* (b) *operation* are more expensive than those discovered in design. Give *two* other justifications for using formal methods in this case. [4]

(ii) Give *two* concrete examples in each case of how a specification produced by traditional methods may be (a) *incomplete*, (b) *ambiguous*, (c) *inconsistent*. Also give *one* example of (d) an *unstated assumption*. [7]

(iii) When modelling the *state* for this application, what *two* main kinds of information should be taken into account? Give *two* examples of detailed pieces of state of each kind which need to be modelled (you need not use formal notation). [6]

(iv) Give three *invariants* that might be used in specifying such a system. [3]