Cardiff School of Computer Science and Informatics
Coursework Assessment Pro-forma

Module Code:	CMT304
Module Title:	Programming Paradigms
Lecturer:	Frank C. Langbein
Assessment Title:	Functional Programming
Assessment Number:	2 of 4
Date Set:	28th November 2022
Submission date and Time:	by 5th May 2023 at 9:30am
Feedback Return Date:	2nd June 2023

If you have been granted an *extension* for extenuating circumstances, then the submission deadline and return date will be 1 week later than that stated above (same time).

This is assignment **two** of a portfolio that will be composed of **four** assignments. This assignment is worth 25% of the total marks available for this module.

If coursework is submitted late (and where there are no extenuating circumstances):

1. If the assessment is submitted no later than 24 hours after the deadline, the mark for the assessment will be capped at the minimum pass mark;
2. If the assessment is submitted more than 24 hours after the deadline, a mark of 0 will be given for the assessment.

Extenuating circumstances (extension or deferral) can *only* be requested using the extenuating circumstances procedure: https://intranet.cardiff.ac.uk/students/study/exams-and-assessment/extenuating-circumstances. Only students with *approved* extensions may use the extenuating circumstances submission deadline – you will receive an *approval* e-mail; this is not the e-mail confirming the extenuating circumstances *submission*. Any coursework submitted after the initial submission deadline without *approved* extenuating circumstances will be treated as late. Note, if you apply for *deferral*, instead of an extension, you will be given the opportunity of a *reassessment* at the next opportunity. You can apply for deferral after an extension request if this is before the (extended) deadline.

By submitting this assignment you are accepting the terms of the following declaration:

I hereby declare that my submission is all my own work, that it has not previously been submitted for assessment and that I have not knowingly allowed it to be copied by another student. I understand that deceiving or attempting to deceive examiners by passing off the work of another writer as one's own is plagiarism. I also understand that plagiarising another's work or knowingly allowing another student to plagiarise from my work is against the University regulations and that doing so will result in loss of marks and possible disciplinary proceedings[1].

[1] https://intranet.cardiff.ac.uk/students/study/exams-and-assessment/academic-integrity/cheating-and-academic-misconduct

Assignment

Consider a small binary image (or 2D array or matrix) that is represented as a list of lists which contains only the numbers 0 or 1, e.g.,

```
[[0,0,0,0,1,1],
 [1,1,1,1,1,0],
 [1,1,0,0,1,0],
 [1,1,0,0,1,1],
 [1,0,1,1,1,1]]
```

We wish to find the number of pixels in the largest connected component of such images (there can of course be more than one component with the same largest number). A connected component is a cluster of pixels that contain the same value and there is a path from each pixel to each other pixel inside that cluster. A path is formed from a start pixel by moving either horizontally (one element left or right in the same inner list) or vertically (one list up or down in the outer list without changing the position in the inner list) to the next pixel until the end pixel is reached (this is 4-pixel connected, i.e. no diagonal movement). The number of elements in the largest connected component for the value 0 in the above example is 4 (among the 4 components). It is 19 for the value 1 (there is only one component).

Task 1: Write an efficient Haskell function

```
nlcc l v
```

that finds the number of elements in the largest connected component of the binary image (list of lists) `l` for the value `v`. Note, there are multiple, more or less efficient algorithms to solve this problem – make sure you clearly document your approach. Also note, you must write a function, not a full program (so no `main`, etc.) and it must have the above name with two arguments (failing to do so may result in 0 marks for this task). Make sure your Haskell code can be compiled/interpreted without errors (otherwise 0 marks may be assigned for this task).

Note that you must write your own code to solve this problem and not just call a library function, or copy code from some other source (independent of plagiarism issues, even if you reference; you only get marks for your own work). You may use the standard libraries listed in the Haskell 2010 language report, but not any other libraries (otherwise the code will be treated as not compilable/interpretable, which may result in 0 marks for this task).

Task 2: Write a short report on functional programming related to the problem:

1. Provide, in up to 300 words, two arguments for and two arguments against using functional programming to solve *this problem*.
2. Discuss, in up to 300 words, whether the functional programming paradigm is suitable for *this problem* or whether another paradigm of your choice is more appropriate, based on your previous arguments.

The word limits are an upper limit, not a target length. Text longer than the word limit for each point may be ignored. Clearly mark each argument in your answer of the first point and indicate whether it is for or against. Only provide two arguments for and against; additional arguments will be ignored (this includes multiple arguments passed as one).

Learning Outcomes Assessed

- Explain the conceptual foundations, evaluate and apply various programming paradigms, such as logic, functional, scripting, filter-based programming, pattern matching and quantum computing, to solve practical problems.
- Discuss and contrast the issues, features, design and concepts of a range of programming paradigms and languages to be able to select a suitable programming paradigm to solve a problem.

Criteria for assessment

Task 1: maximum 50 marks, assessed according to the following scale:

Fail	0	No code has been submitted.
	$1-14$	Code does not run or does not produce valid output for any valid input; little to no relevant documentation.
	$15-24$	Code is valid without syntax errors and creates a valid output for every valid input (or produces a suitable error message for valid cases it cannot process). The output is not a solution, but a suitable attempt to solve the problem is visible. An attempt to document the code has been made.
Pass	$25-29$	Code is valid without syntax errors and creates a valid output for every valid input (or produces a suitable error message for valid cases it cannot process). A suitable attempt to solve the problem has been made, that will often produce the correct output. The attempt has been reasonably documented, but no consideration has been given to optimise the function's performance.
Merit	$30-34$	Code is valid without syntax errors and creates a valid output for every valid input (or produces a suitable error message for valid cases it cannot process). A suitable attempt to solve the problem has been made, that will find the correct output. The attempt has been well documented, stating the idea to solve the problem and how it has been implemented.
Distinction	$35-50$	Code is valid without syntax errors and creates a valid output for every valid input. A suitable attempt to solve the problem has been made, that will find the correct output for all problems, with excellent performance. The attempt has been well documented clearly stating the idea to solve the problem and how it has been implemented. It clearly shows an effort to optimise the program's performance, e.g. by using efficient algorithms, data representations or heuristics.

Task 2: maximum 50 marks, assessed according to the following scale:

Fail	0	No document has been submitted.
	$1-14$	An insufficient number of arguments has been submitted and/or they hardly apply to the functional programming paradigm. At most an incomplete attempt to discuss the suitability of the functional paradigm has been made.
	$15-24$	An insufficient number of arguments has been submitted, but they show some understanding of the functional programming paradigm. An attempt has been made to discuss the suitability of the functional paradigm, but it hardly relates to the paradigm.

Pass	$25-29$	The required number of valid arguments has been submitted. They are generally valid for the functional programming paradigm, but they repeat similar issues, do not consider the specific problem or contain mistakes in the details. A attempt has been made to discuss the suitability of the functional paradigm and some understanding of this paradigm is present.
Merit	$30-34$	The required number of valid arguments has been submitted. They show a clear understanding of the functional programming paradigm and it relates to the problem. The discussion of the suitability of the functional paradigm is well-developed, showing a clear understanding of the issues involved, and indicates the differences to the other chosen paradigm.
Distinction	$35-50$	The required number of valid arguments has been submitted. They show a clear understanding of the functional programming paradigm and the underlying theoretical concepts and/or realisations on programmable machines and how these relate to the problem. The discussion of the suitability of the functional paradigm is well-developed, showing a deep understanding of practical and theoretical issues involved, and clearly discusses concrete differences to the other chosen paradigm.

Feedback and suggestion for future learning

Feedback on your coursework will address the above criteria. Feedback and marks will be returned on 2nd June 2023 via Learning Central. This will be supplemented with individual feedback on request via e-mail.

Submission Instructions

All submissions must be via Learning Central. Upload the following files in a **single zip file**, `[student number].zip`:

Description		Type	Name
Task 1	**Compulsory**	One Haskell source file	`task1.hs`
Task 2	**Compulsory**	One PDF (.pdf) file	`task2.pdf`

Any code submitted will be run on a system equivalent to the Linux laboratory machines and must be submitted as stipulated in the instructions above.

Any deviation from the submission instructions above (including the number and types of files submitted) may result in a mark of zero for the assessment or question part.

All submissions will be compared to each other and checked against other work available on the Internet and elsewhere to identify cases of potential unfair practice.

Staff reserve the right to invite students to a meeting to discuss coursework submissions.

Support for Assessment

Questions about the assessment can be asked on https://stackoverflow.com/c/comsc/ and should be tagged `CMT304fp`; or you can ask your question in or after any synchronous session with the assessment setter.