**Functional Programming**

**5CM524**

****

Assessment Brief

Prof Stephan Reiff-Marganiec

# Module Leader

* Prof Stephan Reiff-Marganiec
* s.reiff-marganiec@derby.ac.uk
* As Head of School, I have a diary with very irregular commitments. The best opportunity to catch me will be before and after the lectures, but you can always try to find me in MS141 or email me/ reach out on teams and I will make arrangements to meet you.

# Key dates and details

| Assessment Detail | Assessment Information |
| --- | --- |
| Assessment Type: | Individual  Portfolio consisting of programming artefacts |
| Assessment Weighting: | 100% (or P/F) |
| Word count/Length: | 15 hours (estimated time required to complete the 2 programming exercises). |
| Learning Outcomes: | 1, 2 |
| Submission Method: | Blackboard Assignment |
| Submission Date: | The portfolio consists of a number of components with deadlines as follows:  Programming exercise 1: 12:00 Noon UK time, 10/01/2025 (40%)  Programming exercise 2: 12:00 Noon UK time, 16/05/2025 (40%)  In addition labs in the following weeks will have a grade attached: weeks of 23/9/24, 18/11/24, 10/2/25, 24/2/25, 7/4/25. (Average of best 4; 20%) |
| Provisional Feedback Release Date: | You will receive feedback after submission of each component as follows:  Programming exercise 1: 31/01/2025  Programming exercise 2: 06/06/2025  Lab feedback will be in the lab session. |

# Description of the assessment

Your assessment consists of a portfolio of programming artefacts. Specifically, there are two programming assignments that you will complete in your own time and submit through Blackboard and there will be 5 labs that are assessed (only the 4 best labs will count towards the grade).

The overall portfolio evidences your growing understanding of using functional programming in several common frameworks and environments to address real-world problems. Details of what is required can be found under Assessment Content further down in this document.

We will discuss the overall assessment approach in the first lecture and the tasks for the programming exercises in the 3rd week into each semester. I am always happy to take questions on the assessment brief before or after classes but the team will not be able to tell you that what you are doing is right or that it will gain full marks.

**Submitted programming assessments:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Portfolio Component** | **Deadline** | **Weigth** | **Feedback** |
| Programming exercise 1 | 12:00 Noon UK time, 10/01/2025 | 40% | 31/01/2025 |
| Programming exercise 2 | 12:00 Noon UK time, 16/05/2025 | 40% | 06/06/2025 |

**Assessed labs:**

The labs in the following weeks will form part of the ongoing assessment component of the portfolio. The grades for the best 4 labs will be averaged and contribute 20% of your portfolio grade:

|  |  |  |
| --- | --- | --- |
| **Lab in week of:** | **Focus of lab:** | **Submission to Blackboard due:** |
| 23/09/24 | Functions in Excel | 27/09/2024 |
| 18/11/24 | Recursive Functions in Haskell | 22/11/2024 |
| 10/2/25 | Countdown problem in Haskell | 14/02/2025 |
| 24/2/25 | Advanced Functions in Excel | 28/02/2025 |
| 07/4/25 | Functional Programming in Python | 11/04/2024 |

Grades for the labs will be shown on Blackboard within a week of the submission due date.

# Relationship to Programme Assessment Strategy

The assessment in this module builds on your learning from Logic & Discrete Mathematics/ Computational Mathematics as well as Programming in Python from Level 4 and provides a toolset of techniques and approaches that you can employ in distributed systems, AI & Data Analytics modules as well as your independent studies module at Level 6. You might also find some of the learning useful when considering how to address problems presented in the Teamwork module at Level 5.

# Attributes and Skills

The table below identifies a few skills sets and resources that are useful for this module/ assessment and which you will also foster and grow by engaging. Skills are a very personal thing and each and everyone of you will be bringing different skills and approaches to their work and hence find different aspects more challenging. The Develop@Derby resource has a numerous support tools embedded, such as self-learning and reading materials, pointers to resources and pointers to classes; it is available to support you beyond the core module content.

|  | Skills | Links to useful resources |
| --- | --- | --- |
|  | Critical thinking |  |
|  | Communication |  |
|  | Collaboration |  |
| X | Creative problem solving | Problem Identification, Divergent and Convergent thinking, Agility (see e.g. <https://libguides.derby.ac.uk/ld.php?content_id=34620879>) |
| X | Self-direction & planning | <https://libguides.derby.ac.uk/independent-learning>  (esp time Management, being organised and beating procrastination sections in <https://libguides.derby.ac.uk/c.php?g=704238&p=5069255>) |
| X | Numeracy, statistics & financial literacy | <https://libguides.derby.ac.uk/elevate-my-maths> (Elements of D1, D2 and D3 on fundamental numeracy & algebra might be useful) |
| X | Digital | <https://libguides.derby.ac.uk/microsoft-office> (specifically section on Excel)  <https://libguides.derby.ac.uk/GenerativeAI> |
| X | Resilience | <https://libguides.derby.ac.uk/academic-wellbeing>  <https://libguides.derby.ac.uk/c.php?g=703820&p=5065126>  (Growth mindset) |
|  | Adaptability |  |
|  | Leadership & future thinking |  |

# Assessment Content

Please find below the details for each of the two programming exercises as well as some more details on the submission requirements for the assessed labs. The first two briefs contain submission instructions and a description of the tasks to be completed; for the assessed labs only the submission and approach guidance is provided; the actual tasks will be in the respective lab instructions.

**Brief for Programming Exercise 1:**

**Overview:** This component is an individual piece of work to be completed in Haskell and to be submitted by 10/01/2025.

**Tasks:** You are asked to write a parser and evaluator for mathematical expressions using recursive functions in Haskell. You should not use any parsing libraries for solving this task. The subtasks below explain what is expected on each task and how they will contribute to the marks for the assessment. The exercise is graded out of 100%.

Task Ex1.1 Basic Parser

The first task is to write a parser for mathematical expressions, using recursive functions in Haskell. The parser (‘parse’) is a function with the following type: parse :: String -> Tree.

Your parser should be able to parse basic mathematical expressions containing +, -, /, \* and ^ (addition, subtraction, multiplication and division as well as exponentiation) into correct parse trees that respect the usual operator precedence. The expression to be parsed will only contain positive whole numbers.

You can assume that the expression entered is syntactically correct.

A correctly working evaluator will be awarded 30 points. Partial grades will be awarded for attempts that manage to parse the expression but do not reflect operator precedence correctly.

Task Ex1.2: Expression evaluator

The evaluator will take a parse tree and calculate the result of the expression. So executing eval(parse(2+4\*3)) should result in the output 14. The evaluator (‘eval’) is a function with the following type: eval :: Tree -> Double

You can assume that the parse tree contains valid numbers and operators, i.e. that the equation can be evaluated into a number.

A correctly working evaluator will be awarded 30 points. Partial grades will be awarded for attempts that manage to traverse the parse tree but do not work in the right order.

Task Ex1.3: Complex expressions

This task is all about extending your parser and evaluator to handle more complex mathematical expressions:

a. Expressions can contain parentheses (‘(‘ and ‘)’). Note that parentheses change the operator precedence, leading to possibly varying results i.e. 2+4\*3 = 2+(4\*3) = 14 but (2+4)\*3 = 18.

b. Expressions to be parsed can contain negative numbers (i.e. input could be ‘2\*-4’).

c. Expressions to be parsed can contain Real numbers (i.e. numbers with a decimal point; e.g. ‘2.5/0.5’)

Each subtask can unlock an additional 10 marks, if working correctly.

Overall structure and coding standard:

Well-structured and laid out code as well as complete type information for all functions written will be awarded 10 points. Fewer marks will be awarded where type definitions are missing or layout/ structuring are such that the code is hard to follow.

**Submission**: Submissions will be made to the Programming Exercise 1 submission point in Blackboard. You need to submit a zip file of the Haskell programme script(s) that contain your solution. Your submission should include all files required for me to execute the submission in a Haskell environment.

**Brief for Programming Exercise 2:**

**Overview:** This component is an individual piece of work to be completed in Python and to be submitted by 06/06/2025.

**Tasks**: You are required to write code to address 2 common real-world needs. The solutions are to be written in Python using a functional programming style. Solutions that do not use a functional style for the substantial parts of the tasks will not be awarded grades. We will elaborate in more detail in class on what this means but briefly, the parser in the first tasks cannot use a global data structure to which you add in an object-oriented or imperative way and the data processing must make use of concepts such as filter and map. The two problems to be addressed are the parsing of complex JSON structures and a typical section of a data processing pipeline. The exercise is graded out of 100%.

Task Ex2.1: JSON parsing

You need to write a parser using a functional programming style in Python which makes use of recursive functions to parse complex JSON structures as might be returned from web services or be exchanged between programmes. Consider the example data shown at the end of the briefing document (Example Data for Programming Exercise 2) for such a complex type capturing departments in a company and some details of their employees. The parser should store the data into an in-memory tree and you should then print the tree in a depth-first fashion.

You can gain 40 points for a correctly working solution, with partial marks only being awarded if your solution is not recursive and for faults it introduces in the order when storing/ printing the data.

Task Ex2.2: Arbitrary JSON

Extend the code from Ex2.1 to allow for arbitrary nested JSON structures containing a mix of nested levels of objects and arrays. You might wish to carefully think about the tree structure that you are creating to ensure that it captures the required insights from the JSON structure.

You can gain 10 points for a correctly working solution using a tree that captures all information needed to access elements from the JSON structure; partial grades will be awarded for solutions that do not capture all information or that are not completely working.

Task Ex2.3: Data processing pipelines

Consider the following real-world scenario in an IoT context: Data has been gathered for PV installations, with data sets publicly available at https://data.london.gov.uk/dataset/photovoltaic--pv--solar-panel-energy-generation-data. A subset of that data is being made available as part of the assessment pages on Blackboard (specifically the file ‘EXPORT HourlyData – Customer Endpoints.csv’ which records data gathered from customer endpoints, such as the voltage and currency generated and power generated). [NB the File PV Tool – Dataset Notes provides some insight into what fields are stored and is provided for your interest.]

The solution you need to develop is a Python programme that can read the file and provide three values (the average of the Voltage, the average of the Current and the total Power) for a specific Substation on a specific day. The extraction of the values and aggregation of the relevant values into the required output should be solved by providing specific Python functions using ideas of map, filter and reduce that you would have seen in the course. The columns in the csv file that are of relevance to solving the task are ‘Substation’ and ‘t\_date’ as well as V\_MIN\_Filtered, V\_MAX\_Filtered, I\_GEN\_MIN\_Filtered, I\_GEN\_MAX\_filteerd and P\_GEN\_MIN.

You can gain 40 marks for correctly working solutions using map, filter and reduce functions in Python as required. 30 marks are reserved for the implementation of the required functions and 10 marks for the overall programme that loads the file, reads input and uses the functions to create the output. Partial evaluations or solutions that do not use the right functions will only gain partial credit.

Overall structure and coding standard:

Well-structured, documented and laid out code will be awarded 10 points. Fewer marks will be awarded where layout/ structuring are such that the code is hard to follow and commentary does not help with understanding of code.

**Submission**: Submissions will be made to the Programming Exercise 2 submission point in Blackboard and will consist of a zip file of the Python programme script(s) that contain your solution. Your submission should include all files required for me to execute the submission in a Python environment.

**Instructions for Assessed labs:**

You will work on a brief in each lab in the course which will be provided through Blackboard in the respective week folder where the lab takes place (i.e. you would find the lab briefing for the week of 23rd September in ‘Week 1’). Note that some lab instructions might span several lab sessions. The module handbook contains a grid mapping content for each week as well as mapping the teaching week to the calendar week.

Each lab will provide you with some tasks to get you started, which will have more detailed explanations and then move you to apply the ideas, concepts and techniques learned to additional problems.

For the 5 identified labs in which you will be assessed, you will work through your lab as normal. A number of tasks on the sheet will be indicated as ‘assessed’. A staff member will look at what you have done as you work through the material and towards the end of the session you will show your work to a staff member when they will complete an assessment sheet (a tick list) highlighting how far you managed to complete work. There will be three levels against each assessed lab task: completed successfully, seriously attempted but not working correctly and not attempted. A serious attempt would mean that you have tried to achieve a complete solution and have worked on resolving matters that were not working correctly (as opposed to a non-serious attempt where you might have written a couple of lines of code in a somewhat thoughtless manner). You will be asked to take a photo of the completed sheet, which you will need to upload into Blackboard by the end of the respective week together with a zip file of the version of the files/code that you did show to the marker. The ‘post session tasks’ on Blackboard will remind you of this.

A grade will be returned through Blackboard by the end of the week following submission which will be calculated from the submitted form. The calculation will be based on awarding 2 points for a seriously attempted task, and 3 points for a successfully completed task and grades will be shown as percentage.

# Assessment Rubric

A detailed point based scheme is attached to the task descriptions. You will see that completing Task 1 in each of the assessments to a working standard with good coding practice will achieve a pass grade. As you complete more tasks you will gain further points that can secure the grade band (in case the initial task is less perfect) or allow you to achieve the higher grade bands.

For assessed labs you will typically be asked to complete 3 graded items; grades returned will be mapped into a percentage value. To pass a lab you will need to have made serious attempts at completing at least 2 of these graded items.

Note that the above looks at gaining pass grades in each component of the portfolio; the grade for the portfolio will be the weighted average of 3 components (the average of the 4 best labs out of 5 and the grades gained for each of the programming exercises) and hence you can balance poor performance in some component by stronger performance in others.

# Anonymous Marking

**Submissions in Turnitin and Blackboard**

You must submit your work using your **student number** to identify yourself, not your name. You must not use your name in the text of the work at any point. When you submit your work in Turnitin you must submit your student number within the assignment document and in the *Submission title* field in Turnitin. [Guidance](https://libguides.derby.ac.uk/c.php?g=708630&p=5110269) is available showing how to do this.

**In-Lab assessment**

Components assessed in the lab are not anonymous as you will present your work to a member of staff. The member of staff will use a marking sheet which they will complete and sign as you present your work. The marking sheet will only contain your student number as an identifier. You will be asked to take a photo of the completed sheet and you need to upload the photo to the respective submission point in Blackboard as your submission.

# Assessment Regulations

The [University’s regulations, policies and procedures](https://www.derby.ac.uk/about/academic-regulations/) for students define the framework within which teaching and assessment are conducted. Please make sure you are familiar with these regulations, policies and procedures.

# Example Data for Programming Exercise 2

Consider for example the following JSON structure:

{

"accounting" : [ {

"firstName" : "John",

"lastName" : "Doe",

"age" : 23 ,

"address": {

"street": "123 Main St",

"zip": "12345"}

},

"sales" : [ {

"firstName" : "Sally",

"lastName" : "Green",

"age" : 27,

"address": {

"street": "456 Main St",

"zip": "23456"}

},{

"firstName" : "Jim",

"lastName" : "Galley",

"age" : 41 ,

“address": {

"street": "789 Main St",

"zip": "34567"}

} ]

}