University of Aberdeen – Department of Computing Science
Modern Programming Languages – CS2510
In-Course Individual Assessment

This is the specification of your individual in-course assessment for CS2510 (Modern Programming Languages). Each item is worth a number of marks and these are indicated next to each item.

What you should hand in: You should hand in **one single ZIP file with your answers to the items below** clearly indicated, and with any instructions necessary to run your programs. You should upload your ZIP file onto MyAberdeen.

When you should hand it in: The deadline for submitting this assessment is **Friday 31st March 11.59PM**.

Lateness penalties: Submissions up to one day late attract a penalty of 10%; submissions up to 1 week late attract a penalty of 25%; work handed in more than 1 week late is marked/returned but counted as a "No Paper"

Plagiarism is a serious academic offence. If you haven't already done so, you should familiarise yourself with the University plagiarism pages available here.

Important: these exercises **are not aimed** at testing your ability to find answers on the Internet, but to assess your ability to come up with your own solutions.

This is an **individual assessment**; your submission will be checked for similarities among submissions of your classmates. We will investigate matters if we have reasons to believe there was any collusion to share answers

This assessment aims at assessing your ability to design and implement solutions using programming languages of different paradigms, specifically, object-oriented (Java) and logic (Prolog) programming. You should develop your own solutions to the programming tasks below, in both programming languages, then write a report comparing the programming effort (process) and the final code (product).

A Scenario: Building Prototypes for an On-line Bookstore

Suppose you are a programmer hired to work for an on-line bookstore; you are tasked with creating prototypes in **Java** and **Prolog** and a comparison report to establish the advantages and disadvantages of each of these prototypes.

The bookstore has a record of their customers and the books they have bought. Here are some examples:

Customer Id: 00245
Books purchased: "Harry Potter and The Philosopher's Stone", "Harry Potter and The Chamber of Secrets", "Harry Potter and The Prisoner of Azkaban"

Customer Id: 98355
Books purchased: "Harry Potter and The Chamber of Secrets", "Harry Potter and The Prisoner of Azkaban", "Harry Potter and The Order of the Phoenix", "Harry Potter and The Half-Blood Prince"

Customer Id: 76644
Books purchased: "Harry Potter and The Philosopher's Stone", "Harry Potter and The Chamber of Secrets", "Harry Potter and The Deathly Hallows", "Harry Potter and The Half-Blood Prince", "Concepts of Programming Languages"

Customer Id: 56779
Books purchased: "Harry Potter and The Chamber of Secrets", "Harry Potter and The Deathly Hallows", "Concepts of Programming Languages", "Harry Potter and The Philosopher's Stone",

In your solutions:

- You should NOT assume you know how many customers there are or how many books they have; your solutions should work for *any* number of customers each with *possibly many* books.
- You should NOT assume that the customers or their books are in any particular order. Your solutions should work for customers and books in any order.

Part 1: Object-oriented programming

For this part, you must use Java (https://www.java.com/en/). Your solutions should run on the standard Java platform and you must provide all the code needed for your solutions to execute.

1.1. Propose a representation for the bookstore information. Use your representation to store (as a test case) the sample records introduced in the scenario above. Create 2 other test cases using your representation. These can be "hard-coded" into your solution, that is, there is no need to use files, or input from keyboard. [10 Marks]

1.2. Implement a solution, using the representation of item 1, to the task of calculating how many books each customer has bought. [15 Marks]

1.3. Implement a solution, using the representation of item 1, to the task of calculating how many customers have bought each book. Your solution can only rely on the records of customers and the books they bought, and *it should not* make use of a record of all books in the shop. [15 Marks]

Part 2: Logic programming

For this part, you must use SWI Prolog (http://www.swi-prolog.org/). Your solutions should run on the standard SWI interpreter and you must provide all the code needed for your solutions to execute. The items below are the same as the ones above, but in this part you must write your solutions in Prolog.

2.1. Propose a representation for the bookstore information. Use your representation to store (as a test case) the sample records introduced in the scenario above. Create 2 other test cases using your representation. These can be "hard-coded" into your solution, that is, there is no need to use files, or input from keyboard. [10 Marks]

2.2. Implement a solution, using the representation of item 1, to the task of calculating how many books each customer has bought. [15 Marks]

2.3. Implement a solution, using the representation of item 1, to the task of calculating how many customers have bought each book. Your solution can only rely on the records of customers and the books they bought, and *it should not* make use of a record of all books in the shop. [15 Marks]

Part 3: Evaluation of Programming Languages used

3.1. For this part you should write up to 3 pages (approximately 1,500 words[1]) comparing ***your experience*** with the two programming languages used, in terms of their readability, writeability and reliability, and the following characteristics:

a) Simplicity
b) Orthogonality
c) Data types
d) Syntax design
e) Support for abstraction
f) Expressivity
g) Type checking
h) Exception handling

[20 Marks]

[1] https://wordcounter.net/words-per-page