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

Module Code:	CMT304
Module Title:	Programming Paradigms
Lecturer:	Víctor Gutiérrez-Basulto
Assessment Title:	Part 1: Logic Programming
Assessment Number:	1 of 4
Date Set:	30th November 2020
Submission date and Time:	24th May 2021 at 9:30am
Return Date:	14th June 2021

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.

Your submission must include the official Coursework Submission Cover sheet, which can be found here:
`https://docs.cs.cf.ac.uk/downloads/coursework/Coversheet.pdf`

Submission Instructions

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

Description		Type	Name
Cover Sheet	**Compulsory**	One PDF (.pdf) file	`[student number].pdf`

Task 1	**Compulsory**	Four source files	`problem_encoding.lp`
			`problem_instance1.lp`
			`problem_instance2.lp`
			`problem_instance3.lp`

Task 2	**Compulsory**	One PDF (.pdf) file	`task2.pdf`

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

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

Your submissions will be checked for plagiarism. Your work must be your own and you must independently solve the problem and submit your own solution. Any other material or sources of information you use must be referenced. Code and text you submit will be compared with other submissions and various other sources on and off the Internet. Any substantial similarities of you submission to unreferenced work or material not created by yourself will be subject to academic misconduct procedures. Marks will only be assigned for work you have done yourself (incl. finding and discussing material from references, but not the referenced work; there are no marks for code copied from elsewhere, but for either writing your own code or integrating and adapting code that you have not written).

Background

This is assignment **one** of a portfolio that will be composed of **four** assignments. Each of the four assignments is worth 25% , summing up to 100% of the total marks available for this module.

Assignment

Consider the following situation:

> The public works division in a region has the responsibility to subcontract work to private companies. There are several types of tasks. Each task is carried out by a team, but each team is capable of carrying out all different types of tasks. The region is divided into districts, and the amount of tasks to be done in each district is known. In particular, the following information is available:
>
> - The region is divided into n districts.
> - There are m private companies such that $1 \ldots k$ are **experienced** and $k+1 \ldots m$ are **non-experienced**.
> - Each company i has t_i teams available, for all $1 \leq i \leq m$.
> - Each district j requires a_j many teams, for all $1 \leq j \leq n$.
> - The yearly cost of allocating a team from a company i to a district j is (the integer) $c_{i,j}$, for all $1 \leq i \leq m$, $1 \leq j \leq n$.

The goal is to write a logic program for helping the public works division with this process. Using the information above, the program should determine the number of teams from each company to allocate to each district such that the following constraints are satisfied.

- At least one experienced company must be allocated to each district (as precaution in case some difficult task arises in that district).
- Enough teams must be allocated to meet the demand in each district.
- No company can be asked to provide more teams than it still has available.
- The cost must be minimised.

Task 1:

1. Write a logic program in ASP (`problem_encoding.lp`) which finds all solutions to the problem, given n, m, k, t_i, a_j, $c_{i,j}$ for all $1 \leq i \leq m$, $1 \leq j \leq n$. Document your code so the following is clear.

   (a) How it should be used.

   (b) What the approach to solving the problem is. In particular, you need to explain **what** each rule achieves and **how** the rule achieves it.

   Include your name and student id in the comments.

2. Write three problem instances (`problem_instance`i`.lp`, for all $i \in \{1, 2, 3\}$) to test your program. Document your code so it is clear what the instance is modeling.

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

1. Provide, in up to 300 words, an analysis of the design and functioning of your program in terms of the Guess-and-Test modeling methodology.

2. Provide, in up to 300 words, two arguments for and two arguments against using logic programming to solve the problem.

The word limits are an upper limit, not a target length. Text longer than the word limit for each point will be ignored. Clearly mark each argument in your answer and indicate if it is for and against. Only provide two arguments for and against; additional arguments will be ignored.

Learning Outcomes Assessed

- Evaluate and apply the logic programming paradigm to solve a given problem.
- Discuss and contrast the issues, features, design and concepts of logic programming.
- Explain the conceptual foundations of logic programming.

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). Even if the output is not a solution, 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 find at least one solution (if there is any). The attempt has been reasonably documented, but no consideration has been given to optimise the program'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 all solutions (if there are any). The attempt has been well documented.
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 all solutions (if there are any) for all problems, with excellent performance. The attempt has been well documented and clearly shows an effort to optimise the program's performance, e.g. by using efficient algorithms and data representations and also some 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 logic programming paradigm. At most an incomplete attempt to analyse the design and functioning of the program has been made.
	$15 - 24$	An insufficient number of arguments has been submitted, but they show some understanding of the logic programming paradigm. An attempt has been made to analyse the design and functioning of the program.
Pass	$25 - 29$	The required number of valid arguments has been submitted. They are generally valid for the logic programming paradigm, but they repeat similar issues, do not consider the specific problem or contain mistakes in the details. A suitable attempt has been made to analyse the design and functioning of the program.
Merit	$30 - 34$	The required number of valid arguments has been submitted. They show a clear understanding of the logic programming paradigm and how these relate to the problem. The analysis of the design and functioning of the program is well-developed, showing a clear understanding of the Guess-and-Test methodology.
Distinction	$35 - 50$	The required number of valid arguments has been submitted. They show a clear understanding of the logic programming paradigm and the underlying theoretical concepts and/or realisations on programmable machines and how these relate to the problem. The analysis of the design and functioning of the program shows a clear understanding of the Guess-and-Test methodology and shows an understanding of related performance issues.

Feedback and suggestion for future learning

Feedback on your coursework will address the above criteria. Feedback and marks will be returned on 14th June 2021 via Learning Central. This will be supplemented with oral feedback on request.