CO3408 Advanced Software Modelling
Assignment 2022/23 (Part 2) – A Specification Spectacular

Deadline: Thursday 6th April 2023, 10.00pm

Part 2 of the assignment contributes 50% of the coursework marks and assesses the following learning outcomes:

1. *Develop and interpret specifications in a formal notation.*
2. *Use and evaluate models in the analysis of a system.*

Background Scenario – A Pantomime of Parking in Preston

Disgraced theme park manager, Dr. Kris Krampus, has fallen on hard times. Having employed a firm of "cowboy" software engineers to help him plan for Christmas at "*100% Lapland Style!"*, he over-spent on present-sorting technology to such an extent that the "Elves' Workshop" made a crippling loss for parent company UCan't plc. The vet's bills for treating the neglected reindeer also ran into thousands of pounds.

To make matters worse, the bunch of undergraduate students Krampus made redundant from their jobs as Elves during their Christmas vacation got drunk, frightened the children with an inappropriately suggestive "elf-dance", and stole most of the decent presents for themselves. The theme park business went into receivership on St. Valentine's day 2023, and Krampus narrowly avoided prosecution when his Patent Present Sorting Machine caused friction burns to the drunken student who fell onto the conveyor belt.

Rather than risking an expensive unfair dismissal tribunal, UCan't decided to "promote" Krampus to a role in another of their business ventures where he could not do as much harm – managing the secure car park at Preston's notorious Wastelands Conference Venue.

A new system is needed to control entry and exit to and from the car park. Dr. Krampus says he knows just the people to help develop the system – You! (Will he ever learn?)

Krampus has learned the following facts about the operation of the car park, and passed them on to you:

- The car park has a fixed-size collection of spaces.
- Entry to the car park is controlled by an automatic barrier.
- There is an area reserved for week-day parking of people who work at the conference centre and pay an extortionate monthly subscription as a percentage of their salary.
    - The number of subscriptions may not exceed the number of reserved spaces.
    - A second automatic barrier controls entry to the reserved area.
    - At weekends, this second barrier remains open, and anyone can park in the reserved area.
- Cars (other than those with a subscription) may enter the car park *only* when the non-reserved area is not "Full" (The car park is deemed to be full when 5 spaces are remaining - these are held back to cater for the idiots who can't park within the lines!)
    - The system needs to maintain a count of how many non-reserved spaces are available.
    - At weekends, ALL spaces count as non-reserved spaces.
- A vehicle registration recognition system will detect whether a car has a subscription, and let them into the car park to drive up to (and through) the second barrier on week-days.
    - The system keeps a record of the registration numbers of cars with a current subscription
- The car park system makes various reports about its current status.
- The car park closes at 11pm each evening, and any cars still parked are towed away to be crushed.

Your task is to develop a **formal specification** for the car park management system in Dafny, and demonstrate that it meets the needs of the town council.

Dafny Implementation **40 Marks** /50

You must develop a Dafny console application in a single .dfy file.

The application should consist of at least one class called CarPark, and a main program which exercises the CarPark class. You may include other simple classes to help if necessary.

To pass, the program must compile and run without errors.

> **NOTE**: Dafny programs will not compile/run if they cannot be verified by the static analyser.
>
> If your program does not compile for **this** reason, you should comment out any pre/post conditions that prevent the program from running, so that you are able to generate the necessary console output

CarPark class (30 marks):

For **high marks**, you should fully and correctly specify any **invariants on the state**, along with **pre- and post-conditions** for the methods or functions (you must decide what parameters are needed) listed below.

For a **pass mark**, you need only specify and implement a simplified version of the system, where there is no reserved area. For such a system, only the first four (**bold**) methods / functions are required.

constructor()	a constructor for the class, setting the car-park up for a new day.
enterCarPark()	to allow any car without a reservation to enter the car park.
leaveCarPark()	to allow any car from any area to leave the car park.
checkAvailability()	to report on the number of non-reserved free spaces currently available.
enterReservedCarPark()	to allow a car with a subscription to enter the car park's reserved area on a weekday, or to enter the car park generally on a weekend day.
makeSubscription()	to allow a car to be registered as having a reserved space when the owner pays the subscription – as long as subscriptions are available.
openReservedArea()	to remove parking restrictions on the reserved spaces (at the weekend).
closeCarPark()	to remove and crush remaining parked cars at closing time.

The specification (and correct implementation) of the above methods will attract between ***2 and 4 Marks each, up to 25 Marks.***

You should represent the state invariant for objects of the CarPark class as a Boolean function as follows:

valid()	evaluates to True when the state invariant(s) hold.

Your representation and specification of the state (including invariants) will attract up to 5 ***Marks.***

MAIN PROGRAM (10 marks)

The main program should initialise a car park for the current day, and run through various scenarios exercising the functions you have specified. The idea is to demonstrate that your specification "works" – exercising the various methods to demonstrate correct behaviour.

You should comment your main program to explain what you are testing at each stage (e.g. // attempting to enter the reserved car park without a subscription).

To help make your main program output more readable, you may consider adding extra functions to the CarPark class such as:

printParkingPlan() to display the car park in rows, indicating the state of each space.

Implementation hints

Try to keep your code simple as far as possible – I am interested in the logic of your pre and post conditions, and invariants. I am not interested in how much of a programming guru you are.

You should explain what you are trying to do as comments in the code – especially the predicates.

- I will mainly be looking at the specification when I mark it, not how you have implemented the methods.
- Your tests (main program) will tell me to what extent the specification has been met.

If you are completely stuck with expressing contracts in Dafny, you will get *some* marks for writing what you would like the pre-and post-conditions to do as English comments before each method. (You should include such comments in any case!)

For a simplified implementation, you could represent the car park spaces as a sequence (or even) array of Booleans, with True representing a space which has been occupied, False one which is available.

Car registrations could be represented as natural numbers (giving a unique ID).

You don't need to go to the extreme of writing a Space class or a Car class.

I DO NOT want you to make a timed simulation (like you made for the previous part of the assignment). When it comes to the second barrier being open at certain times, I am expecting you to put some cars in the car park, then call the OpenReservedArea() method, and then check the behaviour of the car park is as you would expect it to be.

The static verification tool may help you with some aspects of writing the code – but verification of all but the simplest code can be quite tricky. It is something you may wish to discuss in your written answers.

Written Questions **10 Marks** /50

1: Explain your choice of model (data structure) for the state of your car park system.
(200 words max.) ***3 Marks.***

2: Explain why it can be useful to create a formal specification (writing pre- and post-conditions and invariants) before beginning an implementation, and also any drawbacks the approach might have.
(200 words max.) ***3 Marks.***

3: Evaluate the usefulness of the Dafny language, and the Visual Studio Code plugin as a tool to aid the development of such formal specifications. You will gain credit for appropriate citation of literature to support your discussion.
(400 words max.) 4 ***Marks.***

Submission

Please refer to documentation published on the Student Notice-board on Blackboard, regarding plagiarism, and rules for submission of course-work.

Please submit the following electronically via Blackboard:

- Your **Dafny code –** this should be a single .dfy file;
- A single word-processed document containing:
    - A copy of the **console output** after running your simulation;
    - Answers to the written questions.