2 Foundations of Computer Science (asp45)

In this exercise, we will develop a game engine to play a simplified version of the game of Mastermind.

In simplified Mastermind, player A selects a list of \( n \) colours among 3 possible colours: Red, Green and Blue (e.g., [Red; Red; Green; Blue] if \( n = 4 \)). Player B has to guess player A’s list of colours by proposing lists of colours in sequence until she finds the list proposed by player A. Every time player B proposes a list of colours, she gets feedback in the form of a number \( x \). (\( x \) is the number of colours that are in the correct position). For example, if player A’s list is [Red; Red; Green; Blue] and player B guessed [Red; Green; Green; Red], then \( x = 2 \) (the first Red and second Green are at the right positions). Note that \( x \leq n \).

(a) Define a type colour to represent a colour. [2 marks]

(b) Given two colour lists, write a function feedback that returns \( x \). The first argument \( a \) is the list of player A, and the second argument \( b \) is a list of player B. Raise a SizeMismatch exception if the lengths of both lists do not match. You may need to introduce a helper function. [4 marks]

(c) Using currying, define a test function that takes a list proposed by player B and returns \( x \). This function should assume that player A’s list is [Blue; Green; Red]. [2 marks]

(d) What is the type of test in Part (c)? [2 marks]

(e) Write a function generate_lists that generates all possible colour lists of a given length \( n \). The function takes a single argument \( n \). You may use the concatenation operation @ and List.map function. Tip: generate 2 should output \( 3^2 = 9 \) lists. [6 marks]

(f) Given a colour list of player B and a feedback \( x \), write a function valid_lists that takes two arguments \( b \) and \( x \) and returns all possible lists that player A could have chosen (such that they match the feedback given to player B). You may use generate_lists, feedback, List.length and/or List.filter. [4 marks]