

**SCHOOL OF ENGINEERING AND TECHNOLOGY**

**FINAL ASSESSMENT FOR THE BSC (HONS) COMPUTER SCIENCE; YEAR 2 ACADEMIC SESSION : SEPTEMBER 2023**

**PRG2214: Functional Programming Principles Duration: 1 Week**

**Project DEADLINE: Week 14**

**INSTRUCTIONS TO CANDIDATES**

* This assignment will contribute 50% to your final grade.
* This is an individual assignment.

- Late submissions will be awarded 0%

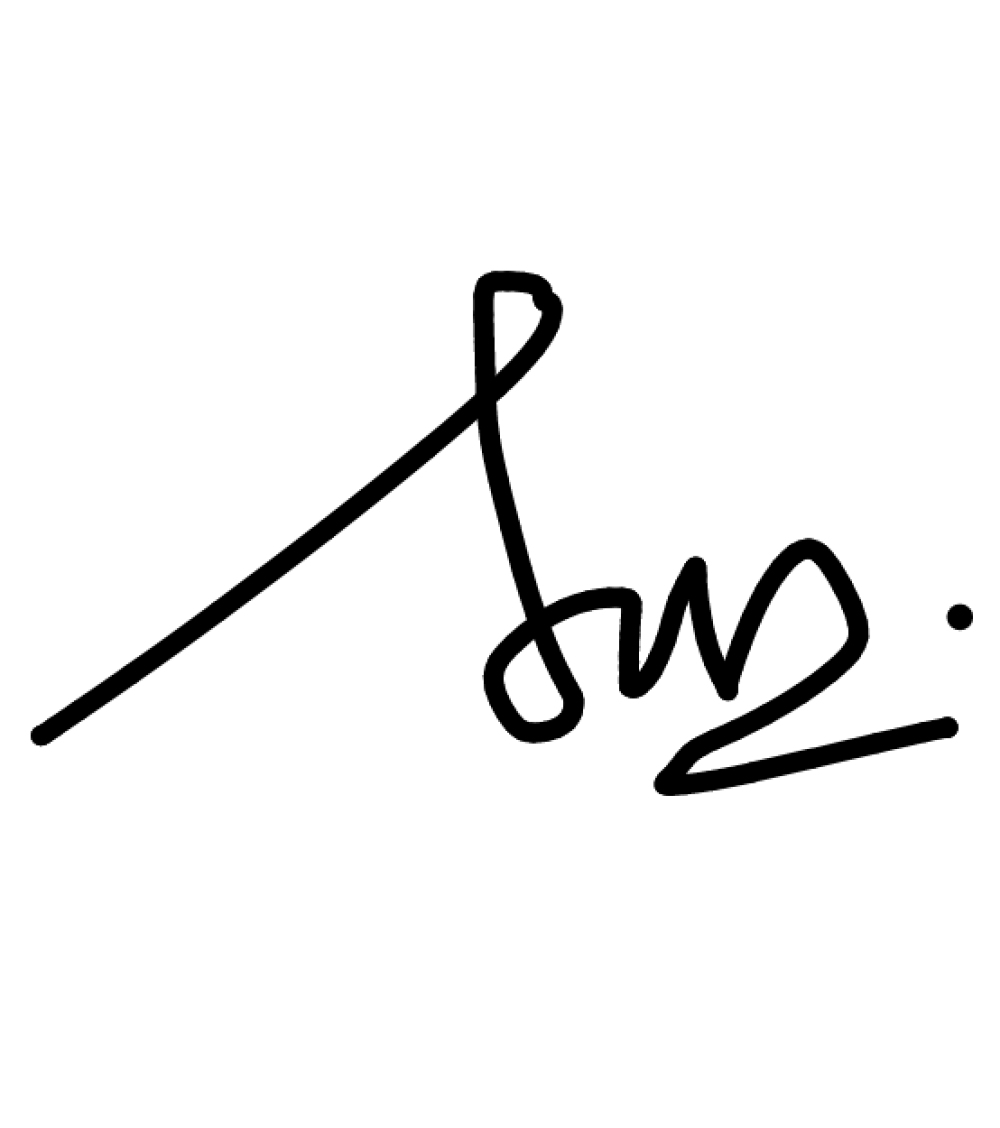
**IMPORTANT**

The University requires students to adhere to submission deadlines for any form of assessment. Penalties are applied in relation to unauthorized late submission of work.

**Lecturer’s Remark** (Use additional sheet if required)

I..B..O...N..G....H...O...N..G....J..U..N. (Name) ..2..2..0..1..8..5..1..9...std. ID received the assignment and read the

comments..................1..7../.1..2../.2..0..2..3 (Signature/date)



**Academic Honesty Acknowledgement**

“I ...B..O...N..G....H...O..N...G....J..U..N (student name). verify that this paper contains entirely my own work. I have

not consulted with any outside person or materials other than what was specified (an interviewee, for example) in the assignment or the syllabus requirements. Further, I have not copied or inadvertently copied ideas, sentences, or paragraphs from another student. I realize the penalties *(refer to page 16, 5.5, Appendix 2, page 44 of the student handbook diploma and undergraduate programme)* for any kind of copying or collaboration on any assignment.”

…............1..7../.1..2../.2..0..2..3 (Student’s signature / Date)

**Table of Contents**

[Introduction 2](#_TOC_250000)

Data Type 3

Function 4

Program Flow & Error Handling 6

Self-Reflection 12

# Introduction

This project involves creating a Recipe Manager application using Haskell, a purely functional programming language. The application serves as a command-line tool for managing culinary recipes, featuring capabilities for adding recipes, searching by ingredients or titles, and generating a comprehensive shopping list.

# Data Types

1. ‘Ingredient’ Data Type:



This data type represents the ingredients used in recipes. Each ingredient has a name and a specified quantity. This data type encapsulates the essential properties of an ingredient, allowing for organized storage and retrieval of ingredient details. Consists of two properties: name (a String representing the ingredient's name) and quantity (also a String, denoting the amount needed). This data type is an instance of show.

1. ‘CookingMethod’ Enumeration:



This data type defines the cooking methods available for recipes. It's an enumerated type, listing methods like Bake, Fry, Boil, and Grill. The use of an enumerated type for cooking methods ensures type safety and clarity in the recipe descriptions. Lists possible methods (Bake, Fry, Boil, Grill) as distinct values. Deriving from Show, Read, and Eq allows for printing, reading from input, and equality comparison of CookingMethod values, respectively.

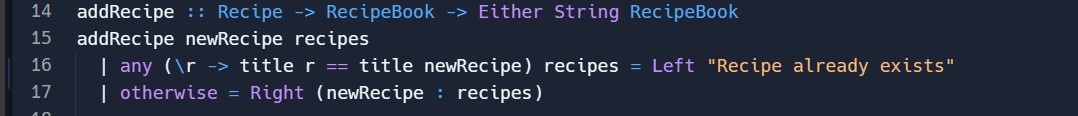
1. ‘Recipe’ Data Type:



This data type represents a complete recipe, including its title, a list of ingredients, and the cooking method. It brings together the ingredient and cooking method data types, providing a structured representation of a recipe. Contains title (a String for the recipe's name), ingredients (a list of Ingredient), and method (a CookingMethod value). The Show derivation enables easy textual display of Recipe instances, incorporating title, ingredients list, and cooking method.

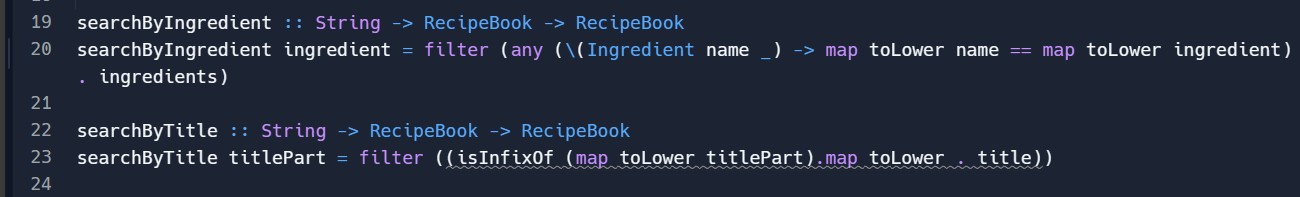
# Functions

1. ‘addRecipe’ Function :



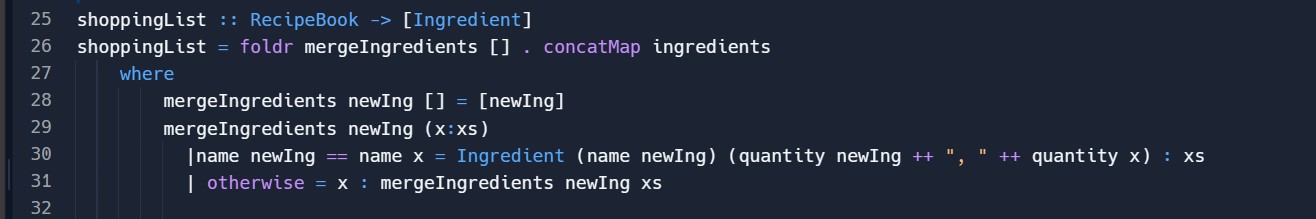
This function adds a new recipe to the recipe book and checks for duplicate titles to prevent redundancy, demonstrating basic error handling. It returns an ‘Either’ type, which can be ‘Left’ with an error message or ‘Right’ with the updated recipe list.

1. Search Functions ( searchByIngredient , searchByTitle ) :



This function allows users to search the recipe book by ingredient or title. It filters the ‘RecipeBook’ (a list of Recipes) to include only those recipes containing a specified ingredient or title. The condition ‘(any (\(Ingredient name \_) -> map toLower name == map toLower ingredient) . ingredients)’ checks if any ingredient in a recipe matches the searched ingredient, ignoring case sensitivity. For the searchByTitle function, I use ‘isInfixOf’ to check if the given title part is a substring of any recipe title.

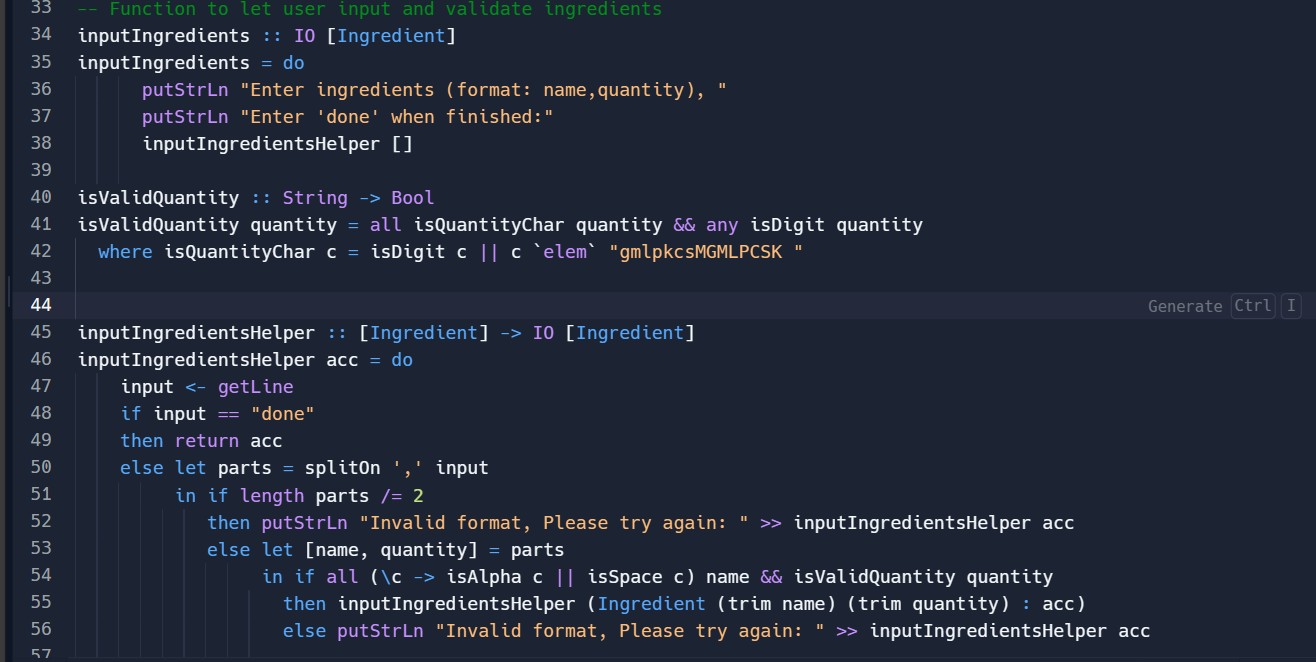
1. ‘shoppingList’ Function:



This function consolidates all ingredients from the recipe book into a single list. This demonstrates the use of ‘foldr’ for list reduction and aggregation. ‘concatMap ingredients’ concatenates all the ingredients from each recipe into a single list. ‘foldr mergeIngredients []’ is then used to reduce this list of ingredients. The ‘mergeIngredients’ function checks if the

current ingredient (newIng) is already in the list. If it is, it merges the quantities; otherwise, it adds the new ingredient to the list.

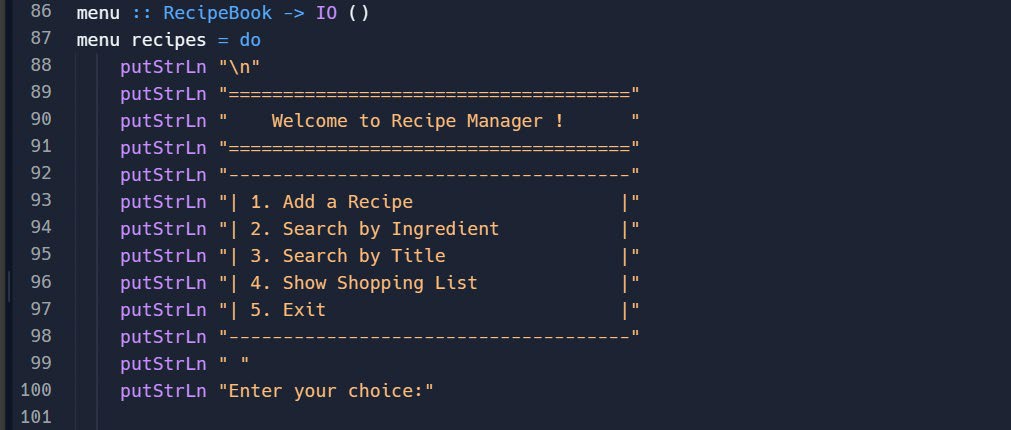
1. User-Input Function (‘inputIngredients’, ‘inputIngredientsHelper’):



The ‘inputIngredients’ function, along with the helper ‘inputIngredientsHelper’, is designed to interactively collect a list of ingredients from the user. ‘inputIngredients’ initiates the process, prompting the user to enter ingredients in a specific format (name,quantity) and indicating the completion of input with ‘done’. ‘isValidQuantity’ is a helper function to validate the quantity format, ensuring it includes digits and valid measurement units (like g, ml, etc.). ‘inputIngredientsHelper’ recursively collects ingredients from user input, validating each entry and splits the input by a comma to separate name and quantity. It also checks if the input format is correct, and the quantity is valid. Continues to accumulate ingredients in the list ‘acc’ until the user types ‘done’.

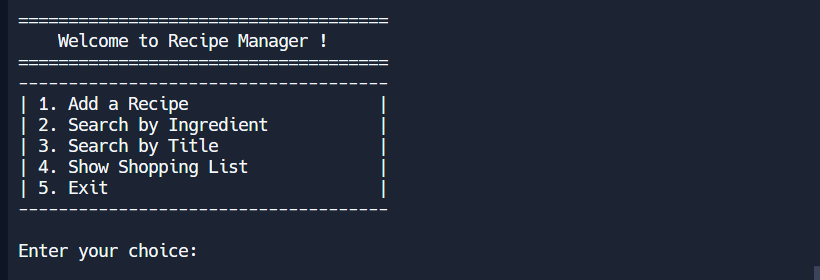
# Program Flow and Error Handling

1. Starting the Program:



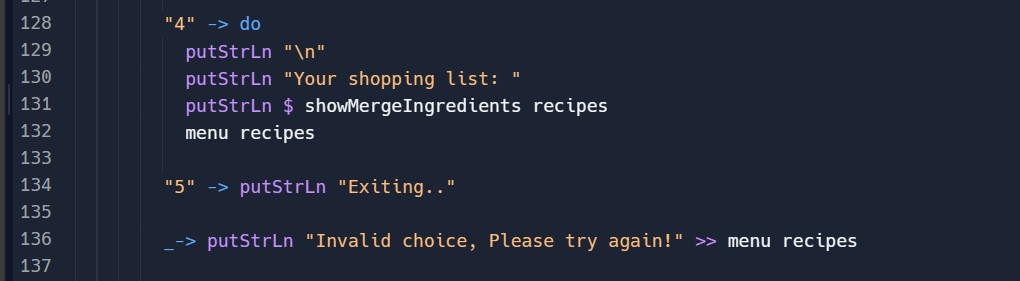
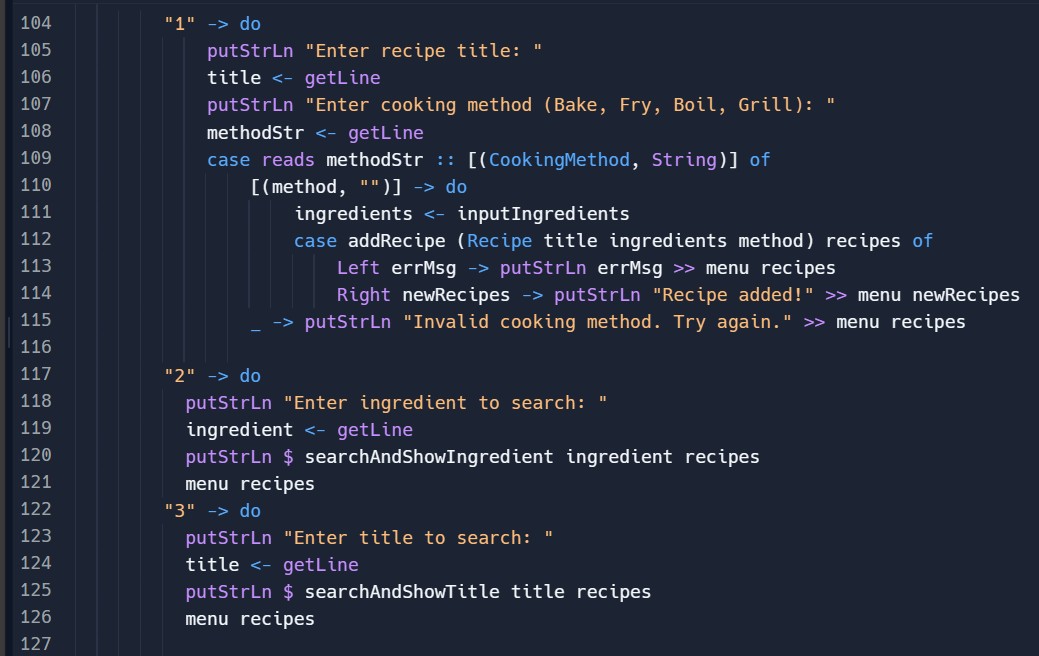
The program begins with the ‘main’ function, which calls ‘menu’ with an initially empty recipe book.

1. Displaying the Menu:

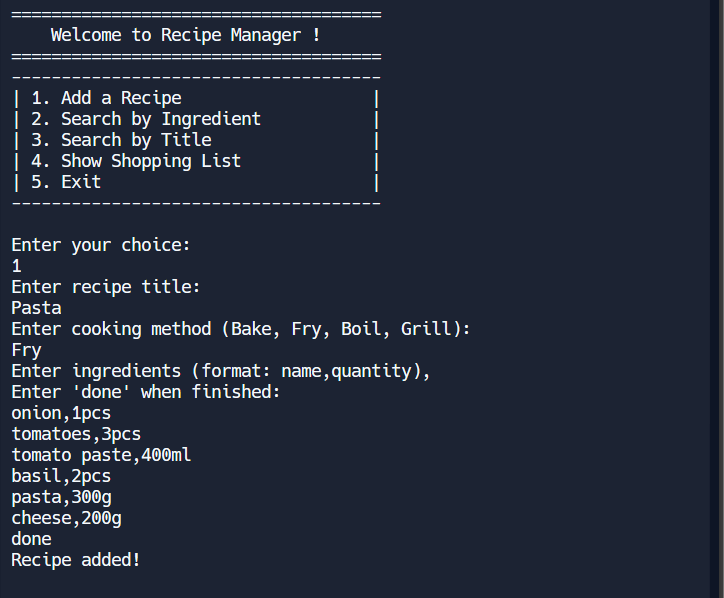


The user is presented with options: Add Recipe, Search by Ingredient, Search by Title, Show Shopping List, and Exit.

1. Handling User Input:



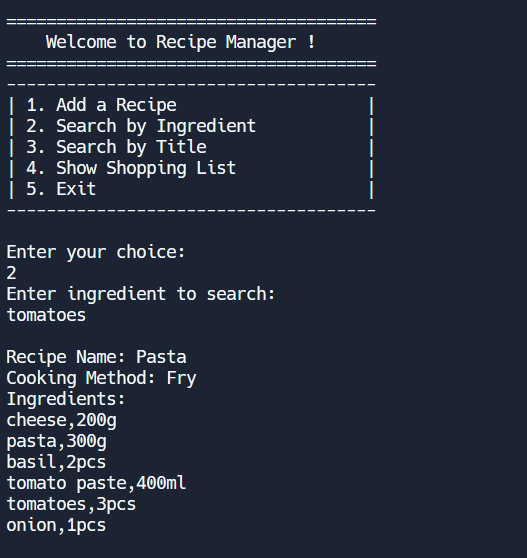
Based on the user’s choice, the program either adds a new recipe, searches for recipes, shows the shopping list, or exits.



For adding a recipe, the user inputs the title, cooking method, and ingredients accordingly.

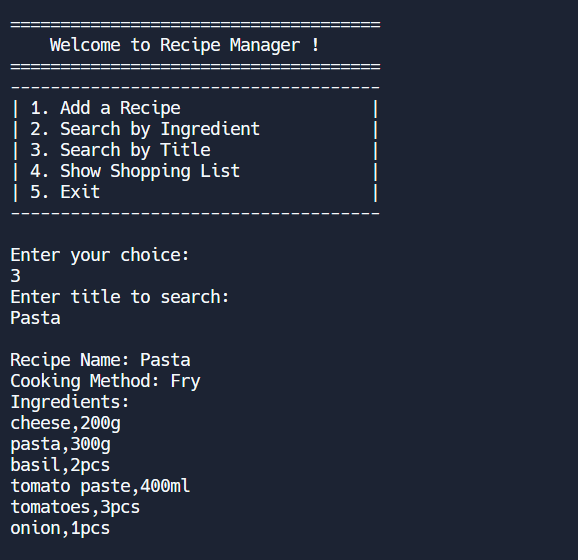
For showcases, I have input the recipe title with ‘pasta’, cooking method with ‘Fry’, and a few ingredients in the correct format (name,quantity). After that, I input ‘done’ when I finish input the ingredients. It will display ‘Recipe added!’ message to tell user the recipe has added.

1. Search Functionality:



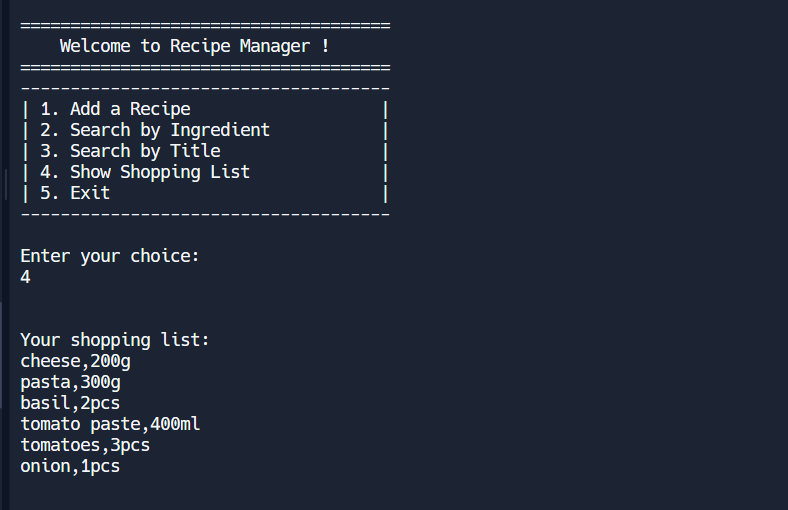
When searching by ingredient, the program filters the recipe book and displays matching recipes.

For the showcases, I input one of the ingredients ‘tomatoes’ that I had input earlier in the ‘Add a Recipe’ function, then the program will search and display the recipe name, cooking method and all the ingredients.

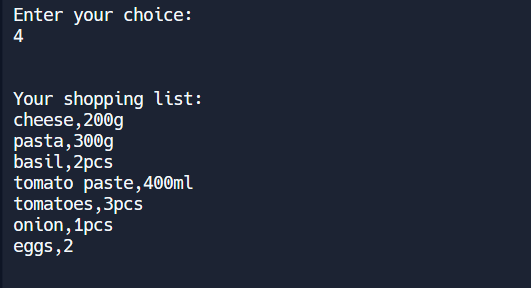
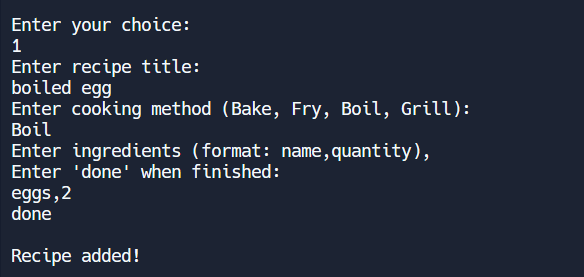


When searching by title, the program filters the recipe book and displays matching recipes. For the showcases, I input the recipe title ‘pasta’ that I had input earlier in the ‘Add a Recipe’ function, then the program will search and display the recipe name, cooking method and all the ingredients.

1. Generating Shopping List:

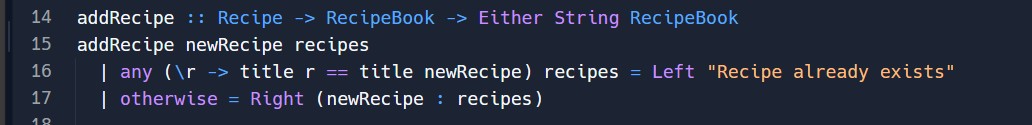


The shopping list function consolidates all ingredients from the recipes into one list, showing combined quantities for duplicate items.

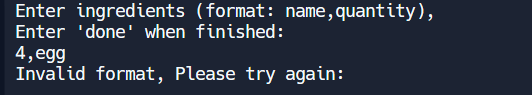
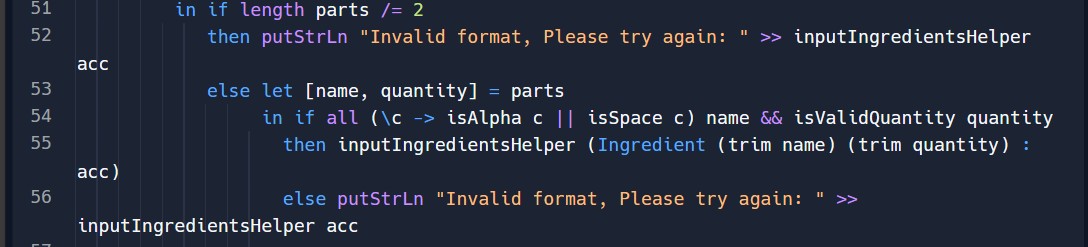


So, for the showcases, I have added another recipe ‘boiled egg’, with ingredients ‘eggs,2’. After adding a new recipe, now I enter ‘4’ again, which representing the ‘Show Shopping List’ function, it is showing combined ingredients that from the recipe ‘pasta’ and ‘boiled egg’ that I have input earlier.

1. Error Handling:



In ‘addRecipe’, errors are handled by checking for duplicate recipe titles and informing the user if a duplicate is found.



In ‘inputIngredients’, the format of the ingredients is validated, and the user is prompted to correct any input errors.

# Personal Reflection

1. Application of Functional Concepts:

I applied Haskell's functional concepts by using pure functions, immutability, and type safety. Functions like `addRecipe` and `searchByIngredient` are designed to be side-effect-free, ensuring predictability and reliability. The use of custom data types and pattern matching further exemplified Haskell's functional approach.

1. Challenges and Solutions:

I encountered challenges in input validation and user interface design. Ensuring robust error handling in `addRecipe` and managing ingredient input required careful consideration. I overcame these by implementing detailed validation checks and clear user prompts.

1. Evaluation of Program:

Strengths: The program's design is strong in its use of functional paradigms, making it reliable and easy to extend. The clear separation of concerns and robust data handling are key strengths.

Weaknesses: The user interface, while functional, is basic and could be enhanced for a better user experience. Also, the program currently lacks advanced features like data persistence or integration with external databases.

REPL.IT Link: [https://replit.com/@prg2214fp082023/Final-Assignment-22018519#Main.hs](https://replit.com/%40prg2214fp082023/Final-Assignment-22018519#Main.hs)