CS4021-FP Assessed Exercise 1: Modelling & Recursion

1 Aims and Objectives

This Assessed Exercise involves writing a medium sized program using Haskell. In this assignment you will be asked to construct a program to generate the receipt for arbitrary shopping lists. Please read the problem scenario presented in §2; attempt the tasks in §3; and answer the questions presented in §4.

To attempt this exercise you should have covered the first four weeks of the MOOC. Please review this online material prior to attempting this exercise.

The main learning objectives of this exercise are to:

- consolidate knowledge about data structures, higher-order functions, and recursion.
- produce a medium sized Haskell program.

2 Problem Description

Mungo's Mangoes is a supermarket in Glasgow. Each item for sale is displayed with a label that shows the item's price and name. Some items are on offer, with the corresponding offer being clearly labelled on the item itself. An item can only be part of one offer at a time. Sometimes items are marked down in price (reduced). Items that are reduced cannot be part of an offer.

The store's checkout machines maintain data on the current offers in store. All offers have the form *Buy One Get One Free* (BOGOF), which means that two items of the same sort can be purchased for the cost of one of those items.

Once items have been scanned in at the checkout, the machine calculates the final receipt. The receipt lists the items bought, together with their full price. Offers are automatically applied and presented in the receipt together with the savings made. Similarly, reduced items are presented in the receipt. The receipt will report the net price of the shopping, together with the total savings made as a result of applying the offers and reductions, and the final price to pay.

3 Tasks

To streamline the work for this exercise, we have broken it down into the following tasks.

3.1 Model the Data

Construct a Haskell module `Items.hs`, that contains data definitions sufficient to model items for sale, and BOGOF offers. Remember to export the data constructors as well as your data types.

```
-- file Items.hs
module Items (...) where
```

```
data Item = ...
data Offer = ...

...
```

3.2 Write the checkout function

In a separate module, define a data structure to represent the final receipt. In this same module write the `checkout` function, that when given a list of items, will calculate the receipt.

```
-- file Receipts.hs
module Receipts (Receipt, checkout) where

data Receipt = ...

checkout :: [Item] -> Receipt
```

3.3 Display the receipt

```
receipt ::= <items> <offers>? <reduced>? <totals> <NL>
items   ::= ititle, (<list index>, <item name>, <price>, <NL>)*, <NL>
ititle  ::= "* Items Purchased", <NL>, NL>
offers  ::= otitle, (bogof)*, <NL>
otitle  ::= "* Offers Applied", <NL>, <NL>
bogof   ::= "+", <list index>, <list index>, <orig price>, <new price>, <NL>
reduced ::= rtitle, (ritem)*, <NL>
rtitle  ::= "* Reduced Items", <NL>, <NL>
ritem   ::= "+", <list index>, <orig price>, <reduced price>, <NL>
totals  ::= <ttitle>, <full>, <osaving>?, <rsaving>?, <total>
full    ::= "+ Full Price ::", <full price>, <NL>
osaving ::= "+ Savings from Offers ::", <offers total>, <NL>
rsaving ::= "+ Savings from Reductions ::", <reductions total>, <NL>
total   ::= "+ Total Price ::", <total price with savings>, <NL>
ttitle  ::= "* Totals", <NL>, <NL>
NL      ::= standard newline character.
```

Figure 1: Format for displaying receipts.

In your module that calculates the receipt, construct a `toString` function to display receipts. Printed receipts must satisfy the eBNF-style encoded grammar in Figure 1.

We realise you may not have encountered formal grammars like this before ... if not, it's basically saying that a receipt has the general format shown in Figure 2.

Each purchased item will be presented in an enumerated list. Items must be displayed in the order they were presented in the shopping basket. If there are no offers/reductions then details about offers/reductions will not be displayed. Prices must be displayed to two decimal places. List indices must be displayed as increasing integers followed by a space. Offers must be applied in order they are found in the input list.

```
* Items Purchased

+ 1 ItemName Price
+ ...

* Offers Applied

+ M N FullPrice HalfPrice
+ ...

* Reduced Items

+ N OrigPrice ReducedPrice
+ ...

* Totals

+ FullPrice :: NN.NN
+ Savings from Offers :: NN.NN
+ Savings from Reductions :: NN.NN
+ Total Price :: NN.NN
```

Figure 2: The general format of the receipt.

4 Questions

In the final part of the exercise you are required to model the following examples. For each example, construct a module that uses your solution to the previous tasks to calculate the cost of the shopping. Then use the provided main module to print the resulting receipts to the terminal standard output. The receipts should be displayed in the order presented below, and each receipt should be surrounded by the following tags: `<start>`; and `<end>`.

Offers This week Mungo's Mangoes has the following offers: (a) BOGOF on apples at 0.50 GBP; and (b) BOGOF on all Cheese blocks at 3.00 GBP.

4.1 Receipt One

Alice went shopping at *Mungo's Mangoes* and bought:

- From the produce section: five 'Apples' at 0.50 GBP per apple; and one 'Watermelon' at 3.00 GBP.
- From the grocery aisle: 'Sumatran Coffee Beans' at 6.00 GBP; and 'Lovage' for 2.59 GBP.
- From the dairy aisle: two blocks of 'Cheese' for 3.00 GBP; and one box of 'Cream Cheese w. Black Pepper', which originally cost 2.50 GBP but is reduced to 1.00 GBP.

4.2 Receipt Two

Bob went shopping at Mungo's Mangoes and bought:

- From the meat counter: 'Pork' (4.00 GBP); and 'Chicken' (4.00 GBP).
- From the dairy aisle: 'Cheese' (3.00 GBP); and 'Mozzarella Cheese'—0.99 GBP.
- From the produce aisle: two 'Apples' (0.50 GBP); and a bag of 'Jaffa Oranges'—3.00 GBP.

Alice went shopping at Mungo's Mangoes and bought:

- From the produce section: five 'Apples' at 0.50 GBP per apple; and one 'Watermelon' at 3.00 GBP.
- From the grocery aisle: one kilogram of 'Sumatran Coffee Beans' at 6.00 GBP; and one jar of 'Lovage' for 2.59 GBP.
- From the dairy aisle: two blocks of 'Cheese' for 3.00 GBP; and one box of 'Cream Cheese w. Black Pepper', which originally cost 2.50 GBP but is reduced to 1.00 GBP.

Mungo's Mangoes has the following offers:

- BOGOF on apples at 0.50 GBP
- BOGOF on all Cheese blocks at 3.00 GBP

(a) A shopping list, and offers.

```
* Items Purchased

+ 1 Apple 0.50
+ 2 Apple 0.50
+ 3 Apple 0.50
+ 4 Apple 0.50
+ 5 Apple 0.50
+ 6 Watermelon 3.00
+ 7 Sumatran Coffee Beans 6.00
+ 8 Lovage 2.59
+ 9 Cheese 3.00
+ 10 Cheese 3.00
+ 11 Cream Cheese w. Black Pepper 2.50

* Offers Applied

+ 1 2 1.00 0.50
+ 3 4 1.00 0.50
+ 9 10 6.00 3.00

* Reduced Items

+ 11 2.50 1.00

* Totals

+ Full Price :: 22.59
+ Savings from Offers :: 4.00
+ Savings from Reductions :: 1.50
+ Total Price :: 17.09
```

(b) The expected receipt.

Figure 3: An example shopping list and receipt.

- Finally he returned to the dairy aisle and picked up another block of 'Cheese' at 3.00 GBP, but only after all the other items had gone through the till (i.e. the two Cheese items are not presented directly one after another).

4.3 Receipt Three

Charlie went shopping at Mungo's Mangoes and bought:

- From the meat counter: 'Pork' that was reduced from 4.00 GBP to 2.00 GBP.
- From the produce aisle: one 'Apple' for 0.50 GBP; and a bunch of 'Bananas' for 0.80 GBP.
- From the dairy: three blocks of 'Cheese', no longer at the full price of 3.00 GBP but all reduced to 1.00 GBP.

5 Supplied Template

You will be supplied with a project skeleton that contains some infrastructure to help us execute your code, and obtain the answers for each question in §4. We will be using the printed output from your submission to ascertain its functional correctness.

Please do not, under any circumstances, alter the supplied project skeleton unless instructed to do so. This helps us to mark your submission more efficiently.

6 Important Information

6.1 Submission Details

Please submit your projects to Moodle using the instructions below. Please make sure you consult Moodle for the submission deadline.

- You are to submit your project as a single **ZIP** archive on moodle.
- Please name the archive using your student number e.g. `1234569x.zip` .
- You must ensure that your project can be run using the provided infrastructure.
- Please do not submit build files or any intermediary files generated by Haskell or other tooling.

If your submission is incorrectly archived, named, or fails to build using the supplied skeleton you may lose marks.

6.2 Marking

The standard mark descriptors will be used and can be found in the student handbook for more information.

- **Clarity of Design**—Was the design of the project suitable for implementing the assignment?
- **Coding Style**—How well was the project presented? Did you make use of idiomatic features?
- **Documentation Coverage**—How complete and useful was the documentation coverage for the project? Were code comments that were given suitable?
- **Functional Correctness**—How sufficient was the implementation w.r.t. to the specification.
- **Submission Quality**—How well was the work submitted?

6.3 Lateness Penalty

Any work submitted later than the advertised deadline on Moodle will be penalised using the standard University penalties. Please consult the student class guide for more information.

6.4 Plagiarism

Information about plagiarism and other examples of academic misconduct can be found in the student class guide and on the University's website.

- `https://www.youtube.com/watch?v=TR7uE81vhK8`
- `http://www.gla.ac.uk/plagiarism/`

6.5 Changelog

List of changes since initial release.

- 2018-10-13: removed backslash from Alice's receipt