Adding New List Operators to App Inventor

Soojin Kim, Wellesley College
Advisor: Professor Franklyn Turbak

The Problem

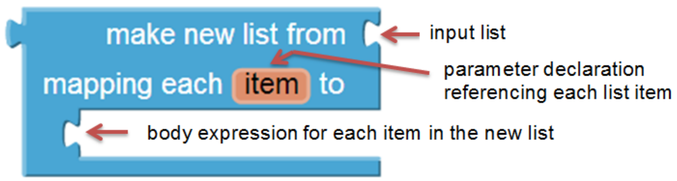
MIT App Inventor 2 (AI2) is a blocks-based programming environment in which users connect puzzle-shaped blocks to build apps for Android devices. AI2 supports a Python-like list data structure typically manipulated with loops, but it lacks important list operations like sorting and reversal. It also lacks standard higher-order list operators like map, filter, and reduce (found in Python as well as many functional languages) that can greatly simplify list manipulation.

The Solution

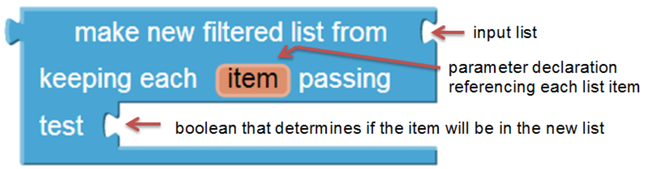
I have extended the implementation of AI2 to include new blocks that map, filter, reduce, sort, and reverse lists. Since AI2 does not have first class functions, many of these blocks incorporate the parameter and body declarations of functional arguments associated with these operators.

In order to implement these blocks, I first created the blocks in Blockly, the JavaScript-based blocks framework for AI2, and then I specified their behavior in Kawa, a version of Scheme that runs on the Java-based Android system. This project is being reviewed for inclusion in the official release of AI2.

New List Operators

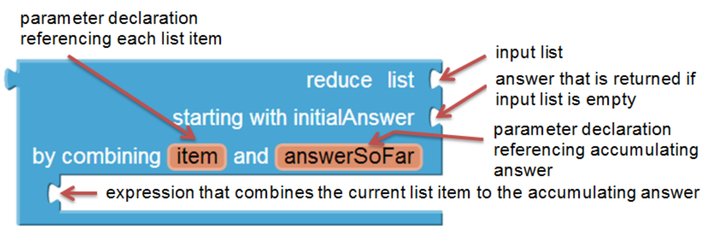


The map block returns a new output list in which every element is the result of performing an operation on the corresponding element of the input list.

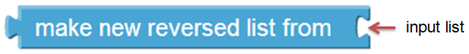


The filter block returns a new output list by keeping only the items in the input list for which the given boolean expression returns true.

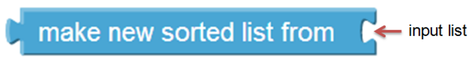
New List Operators (continued)



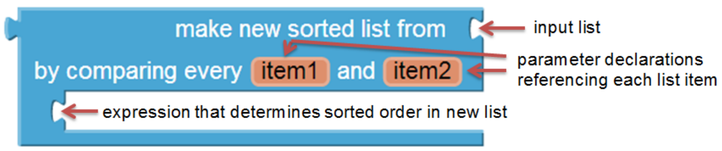
The reduce block combines the items in the input list using the given expression and returns a single value.



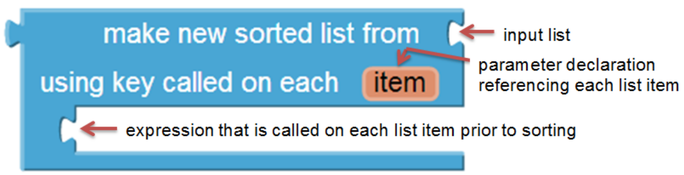
The reversal block returns a new output list whose elements are the elements of the input list in reverse order.



This basic sorting block returns a new output list whose elements are those of the input list sorted in ascending order using a default comparator defined on any two App Inventor values.



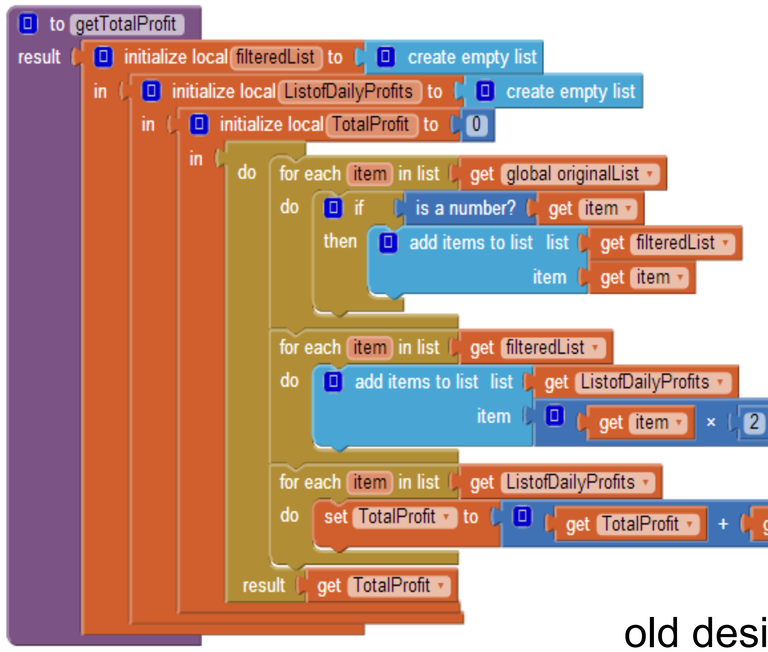
This sorting block returns a new output list whose elements are those of the input list sorted in ascending order according to a comparator defined by a boolean expression that determines if item1 is less than or equal to item2.



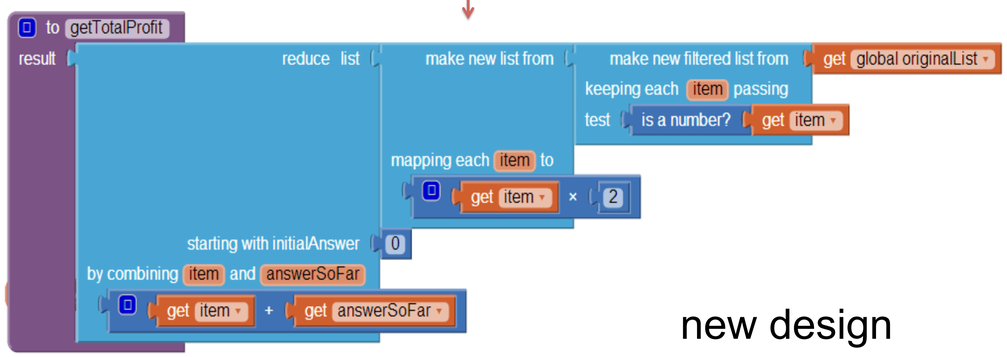
This sorting block returns a new output list whose elements are those of the input list sorted in ascending order using a default comparator on the result of the key expression applied to each item.

Old Design vs. New Design

With these new list operators, users do not have to define their own procedures to manipulate lists. Below, I have included an example involving filtering, mapping and reducing to calculate total profit. I implemented this same example in both the old design (which does not use the new list operators) and the new design. These new list operators enable users to easily process lists with a fewer number of blocks.



old design



new design

Ongoing Work

Each of these new list operators are currently non-destructive (i.e. return a new list rather than modifying the input list), but I am also working on implementing destructive versions and enabling users to select between the two options.