University of Sheffield
Dapartment of Computer Science

COM2001: Advanced Programming Techniques
Semester 1: Functional Programming
Assignment 1: Bags

This exercise counts as 12.5 % of the assessment for the COM2001 module.

1. The Problem

A bag (or multiset) is a collection of items. Each item may occur one or more times in the bag. All the items are of the same type. The order of the items is unimportant. For instance,

- over a 40-game season a football team might score 5 goals in 2 games, 4 goals in 1 game, 3 goals in 4 games, 2 goals in 10 games, 1 goal in 12 games and 0 goals in 11 games;

- a student's assessment profile might comprise 3 A grades, 4 B grades and 2 C grades
.

A bag may be implemented by a list (e.g. [‘A’,’A’,’A’,’B’,’B’,’B,’B’,’C,’C’] for the student grades) but this is clearly inefficient. In this assignment you will implement and test a Haskell module Bags.hs to handle bags.

Bags.hs should contain

- A definition for a polymorphic Haskell datatype **Bag** to represent bags efficiently.
- The following functions

  1. **listToBag** takes a list of items and returns a bag containing exactly those items; the number of occurrences of an item in the list and in the resulting bag is the same.

  2. **bagEqual** takes two bags and returns True if the two bags are equal (i.e., contain the same items and the same number of occurrences of each) and False otherwise.

  3. **bagInsert** takes an item and a bag and returns the bag with the item inserted; bag insertion either adds a single occurrence of a new item to the bag or increases the number of occurrences of an existing item by one.

  4. **bagSum** takes two bags and returns their bag sum; the sum of bags X and Y is a bag which contains all items that occur in X and Y; the number of occurrences of an item is the sum of the number of occurrences in X and Y.

  5. **bagIntersection** takes two bags and returns their bag intersection; the intersection of bags X and Y is a bag which contains all items that occur in both X and Y; the number of occurrences of an item in the intersection is the number in X or in Y, whichever is less.

2. Mark Scheme

	% Credit
Bags Datatype	10
listToBag	10
bagEqual	25
bagInsert	10
bagSum	25
bagIntersection	20

For each function, 60% of the credit is for the quality of the code, 20% for documentation & 20% for testing.

3. What to hand in

Hand in 2 documents

1. Your commented code (.hs file).

2. Your test results for the functions above. These should cover all the logically different cases.

4. How to hand in

Hand in by MOLE

DEADLINE: Midnight on Monday 26th October (Week 5)