COMP 3031 assignment 1
SML programming
Fall 2014

Due: 5PM on 16 October 2014 (Thursday)

Instructions

- There are **five** questions in this assignment. Question 1 counts for one point, Question 2 three points, and Question 3-5 two points each. The total number of points is 10.
- Write your functions exactly the same as defined in the problem description (name, type, and functionality).
- Put your entire solution in a single text file called "*ass1.ml*". In this file, put down your **name, ITSC account, and student ID** as a ***comment*** (surrounded by "(*" and "*)") on the first line.
- Submit your file through the Course Assignment Submission System (CASS) before the deadline: https://course.cse.ust.hk/cass/submit.html
- Instructions on using CASS are available online: http://cssystem.cse.ust.hk/home.php?docbase=UGuides/cass&req_url=UGuides/cass/student.html
- Your submission will be tested under the SML interpreter on a lab machine by issuing the following command:

- *use "ass1.ml";*

- **No** late submissions will be accepted.

Question 1. Hash Table Lookup

We define the following hash table data type to be used throughout Assignment 1:

```
datatype 'a ht =  table of (int * ('a list)) list;
```

Write the following function `lookup` to look up a hash table by a key and return all the values associated with the key. If the table is empty or the key does not exist in the table, return `nil`.

```
val lookup = fn : int -> 'a ht -> 'a list
```

Examples:

```
- lookup 3 (table nil);
val it = [] : ?.X1 list

- lookup 3 (table [(1, [2,3]), (2, [3,4,5]), (3, [4])]);
val it = [4] : int list

- lookup 4 (table [(1, [2,3]), (2, [3,4,5]), (3, [4])]);
val it = [] : int list
```

Question 2. Hash Table Insertion

Write a function `insert` that inserts a (key, value) pair into a hash table. If the key and its associated value already exists in the table, there is no change to the table; If the key exists in the table but the associated value does not, insert the value into the list of values associated with the key; otherwise, insert the key and its associated value into the hash table.

```
val insert = fn : int * ''a -> ''a ht -> ''a ht
```
Note: `''a` stands for a polymorphic type that supports equality test.

Examples:

```
-  val t = table [(1, [2,3]), (2, [3,4,5]), (3, [4])];
val t = table [(1,[2,3]),(2,[3,4,5]),(3,[4])] : int ht

- insert (4,0) t;
val it = table [(1,[2,3]),(2,[3,4,5]),(3,[4]),(4,[0])] : int ht

- insert (3,5) t;
val it = table [(1,[2,3]),(2,[3,4,5]),(3,[5,4])] : int ht

- insert (2,3) t;
val it = table [(1,[2,3]),(2,[3,4,5]),(3,[4])] : int ht

- insert (2,3) (table nil);
val it = table [(2,[3])] : int ht
```

Question 3. Counting the Number of Associated Values for Each Key

Write a function, `valuecount`, that counts the number of values associated with each key for the hash table and returns a list of (key, number of values) pairs.

```
val valuecount = fn : 'a ht -> (int * int) list
```

Examples:

```
-  val t = table [(1, [2,3]), (2, [3,4,5]), (3, [4])];
val t = table [(1,[2,3]),(2,[3,4,5]),(3,[4])] : int ht

- valuecount t;
val it = [(1,2),(2,3),(3,1)] : (int * int) list

- valuecount (table nil);
val it = [] : (int * int) list
```

Question 4. Listing Each Key and Number of Associated Values in the Descending Order of Number of Associated Values

Write a function `hotkeys` that takes a hash table as input and return a list of (key, number of values) pairs sorted by the decreasing order of the number of values for each key.

```
val hotkeys = fn : 'a ht -> (int * int) list
```

Examples:

```
-  val t = table [(1, [2,3]), (2, [3,4,5]), (3, [4])];
val t = table [(1,[2,3]),(2,[3,4,5]),(3,[4])] : int ht

- hotkeys t;
val it = [(2,3),(1,2),(3,1)] : (int * int) list

- hotkeys (table nil);
val it = [] : (int * int) list
```

Question 5. Listing Each Pair of Key and Associated Positive Value

Write a function `positivekvs` for a hash table of integer values to return all pairs of keys and associated positive values:

```
val positivekvs = fn : int ht -> (int * int) list
```

Examples:

```
- val t = table ([(1, [~2,3]), (2, [3,~4,5]), (3, [~4,5,6,~7])]);
val t = table [(1,[~2,3]),(2,[3,~4,5]),(3,[~4,5,6,~7])] : int ht

- positivekvs t;
val it = [(1,3),(2,3),(2,5),(3,5),(3,6)] : (int * int) list

- positivekvs (table nil);
val it = [] : (int * int) list
```