CSE 341 Assignment 1
Basra

January 6, 2006

Due: Thursday, January 12, at 11 pm. No late assignments will be accepted. Submit your assignment using the online turnin link on the course web (when available)

Over the first couple of assignments you will use SML to implement a card game called Basra. For the first assignment, you are to write several SML functions that deals with cards and lists of cards.

Each card should be represented as a pair (*suit*, *rank*), where *suit* is an integer between 1 to 4 corresponding to Spade = 1, Heart = 2, Diamond = 3, Club = 4; and *rank* can be any integer between 1 to 13 (Ace to King).

A list of cards (or "hand") is a list of card pairs. For example, the following list
[(1, 3), (4, 11), (2, 5), (3, 13)]
represents a hand consisting of the four cards: Spade 3, Club Jack, Heart 5, and Diamond King.

This is all you need to know now to implement the first part of the game. The game rules and other details will be included in the next assignment.

Functions to implement

You should implement the following functions in a file called "hw1.sml". You may be able to use some of the functions to help you implement other functions in this assignment.

1. *isValidHand* - given *hand*, which is a list of cards, returns *true* if all cards have valid *suit* and *rank*; return *false* otherwise.
2. *handHunger* - given *hand*, which is a valid list of n cards, return $4 - n$ if n is smaller than 4; if n is greater or equal to 4, return 0.
3. *addToHand* - given *card*, which is a valid card, and *hand*, which is a valid list of cards, return **nil** if *hand* already contains 6 or more cards; else, return a list that has *card* added to *hand.*
4. *removeFromHand* - given *card*, which is a valid card and *hand*, which is a valid list of cards, return a hand which has the specified card removed from it. If the card is not in the hand, the unmodified hand is returned.
5. *removeCardsFromHand* - given *cards* and *hand*, which are two valid lists of cards, remove each card in *cards* from *hand* and return the resulting hand.
6. *earnedScore* - given *hand*, which is a valid list of cards, calculate the accumulating score for *hand*, which is the the sum of the following:
   - each face card (Jack, Queen, King) is worth 2 points
   - each numerical card (Ace to 10) is worth 1 point

Below is a sample execution log:

```
- isValidHand([(1, 3), (2, 4)]);
val it = true : bool
- isValidHand([(5, 3), (1, 14)]);
val it = false : bool
- handHunger([(1, 3), (2, 4)]);
val it = 2 : int
- handHunger([(1, 3), (2, 4), (4, 11), (2, 1), (1, 13)]);
val it = 0 : int
- addToHand((1, 3), [(2, 1), (2, 11)]);
val it = [(1, 3), (2, 1), (2, 11)] : (int * int) list
- addToHand((1, 3), [(3, 3), (2, 4), (4, 11), (2, 1), (1, 13), (3,5)]);
val it = [] : (int * int) list
- removeFromHand((1, 3), [(1, 11), (2, 4), (1, 3)]);
val it = [(1,11),(2,4)] : (int * int) list
- removeCardsFromHand([(2, 4), (2, 1)], [(1, 11), (2, 4), (1, 3)]);
val it = [(1,11),(1,3)] : (int * int) list
- removeCardsFromHand([(2, 4), (2, 1)], [(1, 11), (2, 4), (1, 3)]);
val it = [(1,11),(1,3)] : (int * int) list
- earnedScore([(1, 3), (1, 11), (2, 13)]);
val it = 5 : int;
```

Type Summary

A correct solution will cause these bindings to be printed in the read-eval-print loop:

```
val isValidHand = fn : (int * int) list → bool
val handHunger = fn : 'a list → int
val addToHand = fn : (int * int) * (int * int) list → (int * int) list
val removeFromHand = fn : (int * int) * (int * int) list → (int * int) list
val removeCardsFromHand = fn : (int * int) list * (int * int) list → (int * int) list
val earnedScore = fn : (int * int) list → int
```

Assessment

The homework that you turn in will be graded based on the following:

- Correctness - The code should perform the functions described above
- Style - You should exhibit good style in your coding. Pay particular attention to indentation and commenting. If you are doing something that may be unclear to someone reading the code, you should comment that section and explain what you are doing.
- Do not use material not covered in class. This assignment should be possible using (at most) let bindings and helper functions, and you should not have to use datatypes or case matches.