University of Sheffield, Department of Computer Science

COM2108: Functional Programming
2018 Assignment 2: 8-Off Solitaire part 1

This exercise counts for 25% of the assessment for the COM2108 module

1. Introduction

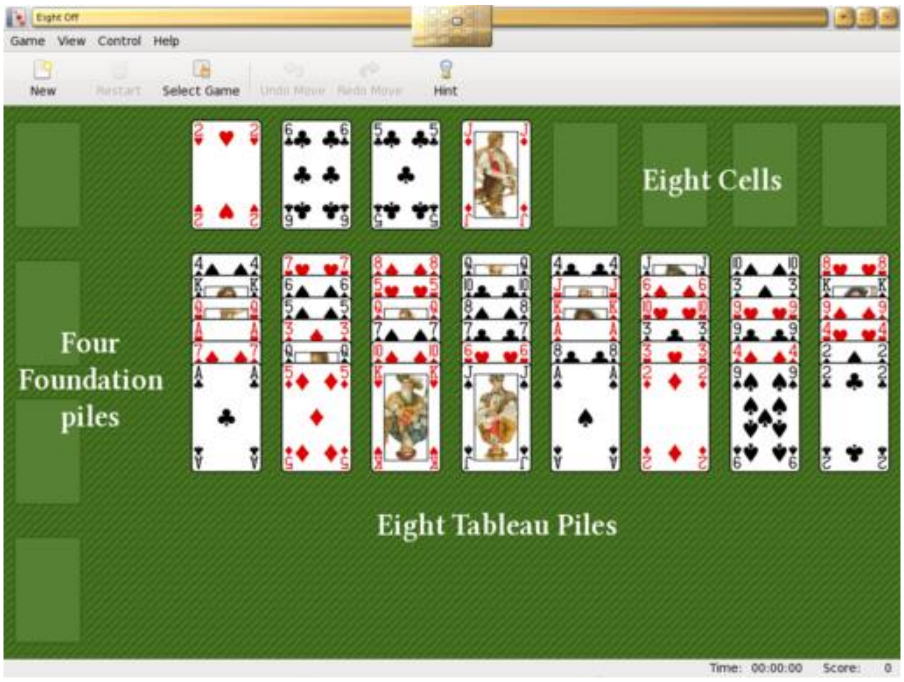
In this assignment you will write code for a Haskell implementation of a Solitaire (Patience) card game. The particular variant of Solitaire is called **8-Off**.

This assignment provides experience of designing list-based and tuple-based data structures in Haskell and algorithms which define the 8-Off operations using recursion, mapping functions and perhaps comprehensions.

In assignment 3 you will go on to write a program to play a game of 8-off solitaire.

2. 8-Off Solitaire

Is played with a normal deck of playing cards. The layout at the start of a game is below



Here is a summary of 8-Off:

Object of the game: Move all the cards to the foundations.

8-Off Solitaire Rules

Foundations (4 piles: complete these piles to win the game)

- Build up in suit from Ace to King (for example, a 2♥ can be played on an Ace♥).
- You must start a Foundation with an Ace

Tableau (8 columns, initially of 6 cards each)

- Build down in suit sequence (for example, a 10♠ can be played only on a Jack♠).
- The top card of each column (the **head**) is available for play to another column, the foundations or the cells.
- In addition, you can move groups of cards from the head of a column to the head of another column if they are in sequence and if there are enough free cells so that the cards could be moved individually. (e.g if one column is 6♥, 7♥, Jack♣… another column is 8♥… and there is at least one space left in the reserve cells you can move the 6♥, 7♥, to leave Jack♣… and 6♥, 7♥, 8♥…)
- If you have <8 non-empty columns, and the head of one of these columns is a King or a King-sequence you can move this King or King-sequence in a new column provided you have sufficient space left in the reserve cells.
- You can move a King from the reserve to an empty column.

Cells (or Reserves; 8 cells)

- These are the "cells". These cells are storage locations for cards being played to the foundations and the tableau.
- The game starts with 4 filled cells.
- There is a maximum of 8 cells.
- Cards in the cells can be moved to the foundations and the tableau.
- Cells can only hold one card.

You should play a few games of 8-off to get the idea. You'll find several web sites for this, or there's a free download on

http://www.solitaire-freecell.com/download_freecell_solitaire.htm

On that page, what you want is **Free FreeCell Solitaire.** The download button is bottom right.

Note that we aren't going to do a simulation with graphics, so the order of the foundations, columns and reserve cells is unimportant.

3. What You Must Do

1. Define Haskell datatypes for
   - A **Suit** (Hearts, Clubs etc)
   - A **Pip** Value (Ace, Two ..King). Aces are low in this game
   - A **Card**
   - A **Deck** of Cards

- **EOBoard**, an 8-off board, consisting of **Foundations**, **Columns** and **Reserve**, which you must also define.

2. Define Haskell constants and utilities as follows

- **pack** – a list of all the cards
- **sCard** – a function which takes a Card and returns the successor card, e.g. the successor of 6♥ is 7♥.
- **pCard** takes a card and returns its predecessor.
- **isAce** is a predicate which is True if a given card is an Ace.
- **isKing** is a similar predicate for a King.

3. Define a function **shuffle**, which returns a shuffled Deck: see §16.3 of the notes
4. Define a function **eODeal**, which returns a shuffled **EOBoard**
5. Define a function **toFoundations** (sometimes called *autoplay*) which takes an **EOBoard** and returns the **EOBoard** obtained by making all the possible moves to the **Foundations.** In the example above, Ace♣ and Ace♥ are available to start foundations. Note that this process is recursive: once Ace♣ has moved, 2♣ will move and that uncovers 2♥.

4. Mark Scheme

	%credit
Data Structures	20
Utilities	10
Shuffle	15
EODeal	15
toFoundations	40

60% of the credit for each section is for coding, 20% for testing and 20% for documentation.

5. What to hand in

Hand in **a single zip archive** containing 2 documents:

1. **Your commented code** as a .hs file, ready to run
2. **Your test results**. You should show that each function specified in section 3 works correctly in each logically different case.

6. How to hand in

Hand in by MOLE

DEADLINE: Midnight Monday 19th November (week 9)