# Graded Assignment Report

## Design

Here is a list of the data structures defined in my solution.

* *Suit* – the suit of a card (Hearts, Clubs, etc)
* *Pip* – the number of a card (Ace, Two, .. ,King)
* *Card* – a *Suit, Pip* tuple
* *SCard* – initially thought to be a “spider solitaire” card, because it is defined as a card that can be either face up or face down, i.e *Card Bool*. It is used in Eight-off Solitaire, although all cards are face up.
* *Deck* – a list of *SCards*
* *Foundations ­*– a list of *SCards* representing the top card of the foundation piles
* *Columns* – a list of *Decks*, representing the columns on an Eight-off board
* *Reserve* – a list of *SCards*
* *Board* – *Foundations*, *Columns* and *Reserve* – that make up a tableau of a solitaire game

This is the design diagram of my implementation of eight-off solitaire in Haskell.

The diagram is not a comprehensive list of all of the functions used in my solution, but rather an overview of the most important ones, on a higher level, and it excludes the helper functions. All of the functions along with their arguments and specifications are specified in the next section.

## Functions

--getCard returns only the (pip,suit) of a card, regardless of visibility

getCard :: SCard -> Card

-- sCard returns successor card

sCard :: SCard -> SCard

--pCard returns predecessor card

pCard :: SCard -> SCard

--check if a card is an Ace

isAce :: SCard -> Bool