Tanay Cansin Dogan
150150119

Functional Programming – Take Home Exam – Part2

Introduction:

In this part we have implemented a customized solitaire game with given customized rules in PFD.

- **cardColor**: checks given card is Black or Red by its Suits (if it's Clubs or Spades it will be Black, rest (Diamon and Hearts) will be Red) via pattern matching
- **cardValue**: returns card value by checking its Rank, If its Ace return 11, if it's Royal its 10 otherwise it's value is equal to it's numerical value
- **removeCard**: Recursively iterate till the end of list, if not found flag an error. If find it, return tail of the list which it takes as input, so can concatenate it with the rest of the list without that specific card.
- **allSameColor**: If list empty it is still same color, If it only has 1 card it is same color. Otherwise (more than 1 card inside) check consecutive 2 card's color, if different than other return false. Otherwise iterate till the end of the list, whenever reached the list with only 1 card ([_] case) it will return True
- **sumCards**: tail recursively sum the values of the cards, the tail recursion handled in sum' helper function. Sum' iterates till the end of the list while adding **cardValue's** of the cards by calling **cardValue** function and carrying the result on acc variable, whenever end of the list reached accumulator returned.
- **Score**: checks held cards are in allSame or not. If allSame return score as preliminary /2, otherwise preliminary. Preliminary score calculated by helper preliminary function as given logic in PDF, as according to compare sum with goal. If sum > goal return 3*(sum-goal) otherwise goal-sum
- **State**: new data type state is defined as Start | End | Continue
- **runGame**: an helper function named **helper** is called, with Start state and empty held list, and card-list as cs and move-list as ms. Helper returns score if it is in End state. If helper takes a empty movelist, empty card-list or sum of the cards is greater than goal, return helper with End state so it can return the score. Otherwise it runs in Continue state recursively and draw a card if move is Draw (card added to held cards) or discard a card (card removed from card-list), **NOT** its time between implementation of the **runGame** and **removeCard** so i forget about removeCard function and implemented a **discarding** helper function for that operation.
- **convertSuit**: take a char and return it as Suit data type
- **convertRank**: take a char (or digit char) and return it as Rank, if it is T returnit as 10 since "10" is a string cant represented as a char it represented via 't'
- **convertCard**: takes 2 Char, char suit s and rank r and return it as Card type by the help of convertSuit and convertRank function
- **convertMove**: takes 'd' or "r _ _" , if its 'd' return Draw, otherwise convert _ _ part to a card and return Discard cardx

- **readMoves**: this function takes user move inputs by the help of recursive helper function readMoves' it takes move list and add each readed move
- **readCards**: same as readMoves function but in order to read Cards.