FIT2102 Assignment 2

Tan Sook Mun

30695759

PickCard

General Idea

For the pickCard, the general idea is that the player will check if the card at the discard pile can form a meld with my hand. It also checks if the card rank is less than 5 and if the card can form a pair (a card with the same rank) then the player will take the card in the discard pile. This condition is so that my AI is heuristic. Because it does not randomly take a card it checks if it satisfies any conditions that will benefit the player.

Function

I created 2 main functions that are actually the conditions for having a heuristic AI.
canMeld :: This function is to check if the card is able to meld with my hand. I concat the card with the hand and makeCombinations of meld. If the card is in any combination of the melds then it can form a Meld with my hand
sameRank :: This function is to check if there is anycard of the same rank so I am able to form future melds. It uses recursion to traverse the code
I also used small functions like (++), (<$>) that I had learn from the tutorials. My functions are also higher order(eg:functions called other functions)

PlayCard

General Idea

When the card is picked now we need to choose which card to discard. So I get the Deadwood and find which Deadwood has the highest value. I will minus the total Deadwood with the deadwood I want to discard because it is the rule of the game. If my overall or total Deadwood(minus the card i want to discard)is less than 10 points then i will call knock. If is 0 then I will call Gin. The player will have a higher chance to win when whoever call knock first. The goal is to get melds faster than the opponent. This strategy also contributes to my heuristic AI.

Functions

The functions I made for this function is maxi and deadwood.
maxi :: Getting the maximum deadwood in the list of Deadwood. Using recursion to traverse the list.
I also used functions form other file like cardPoints. This is so I map the cardpoints to the corresponding Deadwood and sum it up to get the deadwood. My code encourages code reusability.
getDeadwood :: it generate possible combination of the melds and flat map it. Then filter out all the values that are in the melds and in the hand. Then you will get the cards that cannot form melds which are your deadwood. Then use a function toDeadwoodTwo to convert Card to Meld

MakeMelds

General Idea

I generated possible combinations of melds and filter correct melds and convert them all into melds. When I generate my possible combination, I sorted the output in descending order based on the length of the list, Suit then rank. This way you can take the first element as the best meld to choose then slowly filter out the rest. I can do this because when sort by the Suit first I make Straight be a priority because straight can form up to 5 cand meld. Then I filter out the melds that have duplicate/similar cards because you can only have unique cards. Then i get the deadwood and concat it with ++ at the end. the end, you will get the list of all melds

Functions

makeCombinations :: this functions create all possible combinations of the hand and filter out all the correct meld using filterFold and checkMeld
sortSuitRank :: sorts the hand by the Suit then Rank
checkMeld :: checks the length and if it forms a straight or sets. Check meld is basically a function that call multiple functions and makes it into a single function that checks that meds length are correct is a straight or a set. Is a higher-order functions
checkSets :: a recursion that checks that it all has the same rank
checkStraights:: a function that checks that it can form a straight. It will traverse the list and ensure that the first element is a predecessor of the second element.
checkLength:: checks that the length between 3 to 5
toMeld :: is a function that checks what kind of meld it is. First checks the length then it will call the corresponding helper function to convert the card to Meld. If the length is 3 or 4 then it has to check if it is a set or straight.
filterMelds :: Is a function that keeps track of the list of formed Melds and the return list of chosen Melds. The function will traverse each list and check if there is any similar card in the list of formMeld.
anySimmilar :: This function checks if there any card that is similar to the list of card that has been made into meld It checks the length of the list of possible meld then it filters our any card that is similar to the list of Cards in the formed melds.If the length changes then there are similar card if not then is a unique meld

Summary

In general, my AI is a heuristic player. This is because it checks for certain conditions when making a move. It is not random. My AI goal is to get the smallest deadwood as fast as possible and calling knock when it can. It also discards the highest card to lessen the deadwood and make the opponent take a higher card. In my code i had use applicative, monad and functor functions such as <$>, <*>,++, || , \c ->, do block and etc. I also used many functions that I found on hoggle. I also used a lot of functions that I had already made during my tutorial and utilized them. I also have higher-order functions which are functions that are calling other function such as checkMeld. I created a lot of helper functions so my code does not look messy and every evaluation is made into small pure functions. I also try my best to make my function into a single line. Also, Some of my code is in point free form or has been eta reduce.

Improvements

There are many things I would like to improve in my code. I hope to use my memory to store strings of card played. My code does not use memory. My idea of using the memory was to store every card I discarded and when the opponent takes the card in the discarded pile. This way after a few rounds, I can roughly guess the opponent's hand. I had made parsers for cards that I did not use. The parsers are able to parse the card, rank and suit of the card.