Lista de Exercícios 3 – INF05008

Use os nomes de constantes, tipos de dados e funções definidos nas questões, com a grafia solicitada!!!

O jogo de cartas Escova (Escopa) é um jogo de cartas bastante popular no mundo. Se joga com um baralho comum ou espanhol, removendo-se ou as figuras (J, Q e K) ou as cartas 8, 9 10 (neste caso, J vale 8, Q vale 9 e K vale 10). O objetivo do jogo é capturar a maior quantidade de cartas por meio de combinações de cartas no valor de 15 pontos, estas combinações são feitas com uma carta da mão e uma ou mais cartas da mesa. As regras do jogo de escova podem ser encontradas em `https://pt.wikipedia.org/wiki/Escova_(jogo_de_cartas)`.

1. Para podermos construir um programa para jogar ESCOVA, precisamos definir um tipo de dados para representar as cartas do jogo. Cada carta possui um naipe (que pode ser paus, espadas, copas ou ouros) e um valor, que é um número entre 1 e 10. Cada jogador pode ter até 3 cartas na mão, e vamos assumir que na mesa podem haver até 4 cartas (em um jogo real, poder haver mais).

   Construa a definição do conjunto de cartas de um baralho, da mão e da mesa, completando a definição de dados a seguir, e defina 4 constantes de cada um desses tipos (`Carta`, `Mão`,`Mesa`). Incluiremos no conjunto carta uma carta para representar a *carta nula*, com o naipe `nenhum` e o valor zero (esta carta será usada em algumas questões a seguir).

   Convenção usada na disciplina: os nomes dos tipos de dados iniciam com letra maiúscula, mas na definição da estrutura usa-se letra minúscula.

```
;; -----------------
;; TIPO CARTA:
;; -----------------
(define-struct carta (naipe valor))
;; Um elemento do conjunto Carta é
;;   ............     onde
;;   ... : ....., é o naipe da carta, que pode ser "paus", "espadas", "copas", "ouros" ou "nenhum"
;;   ... : Número, .....
;; -----------------
;; TIPO MÃO:
;; -----------------
(define-struct mão (c1 c2 c3))
;; Um elemento do conjunto Mão é
;;   ............     onde
;;   ... : .....
;; ...
;; -----------------
;; TIPO MESA:
;; -----------------
(define-struct mesa (.......))
;; Um elemento do conjunto Mesa é
;;   ............     onde
;;   ... : ....., .....
;;   ...
```

   Para pensar: Note que um jogador pode ter menos do que 3 cartas na mão, mas como usamos aqui uma estrutura *para representar a mão do jogador, que é um* tipo de dado de tamanho fixo*, é necessário definir uma carta (que não existe no baralho) para representar que alguma posição pode estar livre. Para isso estamos usando a* carta nula*.*

2. Desenvolva a função `soma15?` que, dadas 2 cartas, verifica se a soma dos valores delas é 15.

3. Uma das possíveis jogadas em um jogo de escova é jogar uma das cartas de sua mão e remover uma das cartas da mesa. Essa jogada é possível se a soma dos valores da carta da mão e da carta da mesa é 15. Construa a função `soma15` que, dada uma carta da mão e uma mesa, devolve a carta da mesa que será retirada pelo jogador juntamente com a carta da mão (devem somar 15). Se a carta da mão não somar 15 com nenhuma da mesa, devolver a carta nula. A função deve devolver a primeira carta encontrada na ordem das cartas na estrutura mesa.

4. Construa a função `escova?` que, dadas uma carta e uma mesa, verifica se esta jogada resultaria em escova.

5. Defina a função `jogada-escova` que, dadas uma mão e uma mesa, se houver possibilidade de escova, devolve a mensagem (string) `"Escova!"`, caso contrário devolve a mensagem `"Não faço escova."`. Assuma que não há 15 pontos exatos na mesa.

6. Desenvolva a funçãp `seleciona-carta` que, dadas uma mão e uma mesa, devolve a carta selecionada para jogar e a mensagem (string) `"Escova!"`, ou `"Não faço escova."`. A carta selecionada deve ser a primeira da mão que faz escova com a mesa, se houver, ou a primeira carta da mão que não for nula (ignore o fato de uma carta da mão poder somar 15 com as da mesa sem fazer escova). Note que a carta nula não é de fato uma carta, então não deve ser selecionada. Assuma que existe pelo menos uma carta na mão.

   Para pensar: Note que esta função deve devolver uma carta e uma string. Porém, nenhuma função pode devolver dois resultados. Como fazer então? ***Definindo um tipo de dados estruturado para o resultado, que contém uma carta e uma string!***

   Defina um tipo de dado para o resultado e o chame de `Jogada` (usando a convenção descrita acima, o comando deve ser `(define-struct jogada ....)`. Os campos devem se chamar `carta` e `mensagem`, nesta ordem.

7. Faça a função `desenha-carta` que, dada uma carta, gera uma imagem para esta carta, inspirada nas cartas do baralho (comum ou espanhol). Se a entrada for a constante `CARTA-NULA`, devolve a imagem vazia (`empty-image`). *Obs: Não é para copiar uma imagem de uma carta da internet, a ideia é você desenhar a carta usando funções do pacote de imagens. Ver https://docs.racket-lang.org/teachpack/2htdpimage.html*

8. (Desafio - Ponto extra) Desenvolva a função `mostra-jogada` que, dada uma mão e uma mesa, gera uma imagem mostrando a mão, a mesa, e uma mensagem indicando se o jogador fez escova ou não com a carta selecionada. No caso de fazer escova, indique com qual carta. Ver exemplos abaixo (você pode escolher a representação gráfica para as cartas, mesa, etc).

   Dica: Decomponha o problema em problemas menores e construa a solução através da composição das soluções dos problemas menores.Lembre que TODAS as funções devem ter documentação completa.

   Para pensar: O grande desafio é resolver o problema com o código mais legível possível!