Lista de Exercícios 6 (Cap. 10) – INF05008

USE OS NOMES DOS TIPOS DE DADOS E FUNÇÕES DEFINIDOS NAS QUESTÕES (a grafia deve ser exatamente a indicada nas questões). Indique CLARAMENTE no arquivo onde cada questão está sendo resolvida.

Data limite para submissão: 5 de outubro às 23:59

Use as definições de dados a seguir para representar cartas de um baralho de UNO.

```
;; -----------------
;; TIPO CARTA-NUMERO:
;; -----------------
(define-struct carta-numero (cor valor))
;; Um elemento do conjunto Carta-numero é
;;   (make-carta-numero c v)     onde
;;   c : String, é a cor da carta, que pode ser "azul", "verde", "amarelo" ou "vermelho"
;;   v : Número, é o número da carta

;; -----------------
;; TIPO CARTA-ESPECIAL:
;; -----------------
(define-struct carta-especial (cor tipo))
;; Um elemento do conjunto Carta-especial é
;;   (make-carta-especial c t)     onde
;;   c : String, é a cor da carta, que pode ser "azul", "verde", "amarelo",  "vermelho" ou "qualquer"
;;   t : String, é o tipo da carta especial, que pode ser "Compra4", "Compra2", "Inverte", "PulaVez" e "TrocaCor"

;; -----------
;; TIPO CARTA:
;; -----------
;; Um elemento do conjunto Carta pode ser
;; 1. um elemento do conjunto Carta-numero;
;; 2. um elemento do conjunto  Carta-especial
```

1. (Fácil) Defina o tipo de dado `ListaDeCartas`, que contém todas as listas (finitas) de cartas de Uno e dê 4 exemplos de elementos deste tipo de dados. Defina também o tipo de dados `Jogador`, completando a definição a seguir. Para cada jogador deve ser guardado o nome do jogador, a lista de cartas de sua mão e sua pontuação. Dê 2 exemplos do tipo jogador.

   ```
   ;; -----------------
   ;; TIPO JOGADOR:
   ;; -----------------
   (define-struct jogador ( ......................))
   ;; Um elemento do conjunto Jogador é
   ;;   (make-jogador ..............)     onde
   ;;   ...................
   ```

2. (Fácil) Construa a função `insere-carta` que, dados um jogador e uma carta que foi comprada por este jogador, nesta ordem, insere esta carta na lista de cartas do jogador, na frente de todas as outras. Obs.: O resultado é o registro de um jogador.

3. (Média) Construa a função `seleciona-cartas-cor` que, dada uma lista de cartas de Uno e uma cor, nesta ordem, devolve todas as cartas desta cor na lista. Assuma que cartas de cor `"qualquer"` da lista de cartas são compatíveis com qualquer cor, mas se a cor passada como segundo argumento for `"qualquer"` , somente cartas com cor `"qualquer"` podem ser selecionadas.

4. (Difícil) Construa uma função chamada `vencedor` que, dados 2 jogadores, define que ganhou. Para isso, é necessário somar à pontuação de cada jogador os pontos referentes à lista de cartas que ele tem na mão e verificar quem tem a menor pontuação (como o objetivo é ficar com menos cartas, que tem a menor pontuação ganha). A função deve devolver apenas o nome do vencedor. Se os dois tiverem a mesma pontuação, a função devolve a mensagem `"Empate"`.

   Os pontos das cartas de Uno são os seguintes:

   - as cartas numeradas valem o seu número;
   - as cartas "Compra2", "Inverte" e "PulaVez" valem 20;

- as cartas "Compra4" e "TrocaCor" valem 50.

5. (Dificil) Construa a função `joga` que, dados a carta da mesa e um jogador, nesta ordem, escolhe uma carta to jogador para ser jogada, de acordo com as regras do Uno. A carta a ser selecionada deve ser a primeira da lista de cartas da mão do jogador que for compatível com a carta da mesa. Lembrando, as regras para escolha de cartas válidas para jogar são:

   - Uma carta-numero pode ser escolhida se for da mesma cor ou do mesmo numero da carta da mesa;
   - As cartas especiais "Inverte", "Compra2" e "PulaVez" podem ser escolhidas se forem da mesma cor da carta da mesa ou tiverem o mesmo símbolo (mesmo tipo);
   - As cartas especiais "TrocaCor" e "Compra4" sempre podem ser jogadas, e sobre uma carta dessas na mesa qualquer carta da mão pode ser jogada.

   Caso o jogador não tenha nenhuma carta que possa ser jogada, a função deve retornar a mensagem `"Jogada impossível"`.

6. (Fácil) Adapte e complete a função `mostra-jogada` no arquivo `mostra-jogada.rkt` ou `mostra-jogada.txt` para funcionar com suas definições de dados. O objetivo desta função é, dados a carta da mesa e um jogador, nesta ordem, mostrar a jogada escolhida (gerando uma imagem com a jogada, incluindo o nome do jogador). Se você quiser usar suas imagens para as cartas e para mostrar a jogada (desenvolvidas na lista 4), pode modificar essas as funções do arquivo `mostra-jogada` para usá-las. Essas definições devem ser inseridas no arquivo Racket que vocês irão submeter.

7. (Fácil) Construa a função `mostra-jogadas-possíveis` que, dados a carta da mesa e um jogador, nesta ordem, mostra a lista de cartas que poderiam ser jogadas (gera uma imagem com o nome do jogador, as cartas da mão, a carta da mesa, e as cartas possíveis de serem jogadas). Se não for possível jogar nenhuma carta da mão, a saída deve ser a imagem do nome do jogador, a mão, da carta da mesa e a mensagem `"Jogada impossível"`. *Dica: Decomponha o problema em problemas menores e construa a solução através da composição das soluções dos problemas menores, preferencialmente reusando funções definidas nos exercícios anteriores.*