Lista de Exercícios 4 – INF05008

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

Data limite para submissão: 7 de setembro às 23:59

O jogo de cartas UNO foi criado nos Estados Unidos e é um jogo bastante popular no mundo. O baralho de UNO é composto por cartas de quatro cores: verde, amarelo, vermelho e azul. Os números das cartas variam entre 0 e 9. Existem três ações especiais para cada tipo de cor de carta, identificadas como `PulaVez`, `Compra2` e `Inverte`. Há também cartas de ações especiais que não são de uma cor específica (vamos considerar que são de cor preta): `Curinga` e `CuringaCompra4` (não iremos considerar a ação especial de trocar mãos). As regras do jogo de UNO podem ser encontradas em `https://copag.com.br/blog/detalhes/uno`.

1. Constantes são nomes que damos para valores. Podemos definir constantes de qualquer tipo e devemos escolher nomes significativos para as constantes. Duas das grandes vantagens de usarmos constantes são: (i) tornar o código mais legível, pois usamos uma palavra para representar um valor; (ii) tornar o código mais fácil de alterar, pois se quisermos mudar o valor associado à esta constante é necessário apenas alterar o valor no lugar onde a constante foi definida (ao invés de procurar em todo o código os locais onde um determinado valor foi usado). O bom uso de constantes faz com que menos erros sejam cometidos.

   Defina constantes para representar os valores das cartas de ações especiais do UNO. Devem ser definidas as constantes `PULA_VEZ`, `COMPRA2` e `INVERTE`, `CURINGA` e `CURINGA_COMPRA4` (use exatamente esse nomes, com a grafia solicitada). Os valores destas constantes devem ser números que não estejam no intervalo de 0 a 9.

2. Para podermos construir um programa para jogar UNO, precisamos definir um tipo de dados para representar as cartas do jogo. Cada carta possui uma cor (que pode ser verde, amarelo, vermelho, azul ou preto) e um valor, que é um número (os números que representam as cartas de ações especiais são os que você definiu na questão anterior). Construa a definição do conjunto de cartas de um baralho de UNO, completando a definição de dados a seguir, e defina 4 constantes do tipo `Carta`, sendo 2 cartas com valores de 0 a 9 e 2 cartas de ações especiais.

```
;; -----------------
;; TIPO CARTA:
;; -----------------
(define-struct carta (cor valor))
;; Um elemento do conjunto Carta é
;;   .............     onde
;;   ... : ....., é a cor da carta, que pode ser "azul", "verde", "amarelo", "vermelho" ou "preto"
;;                 (ou "livre" - ver questão 4)
;;   ... : Número, .....
```

3. Desenvolva a função `jogada-válida?` que, recebe duas cartas, representando a carta da mesa e uma carta da mão, nesta ordem, e verifica se é possível jogar esta carta da mão de acordo com as regras do jogo UNO (ver site acima). Assuma que, se a carta da mesa for uma carta especial preta, uma carta de qualquer cor pode ser jogada. O resultado deve ser um valor booleano.

4. Em um jogo de UNO, o número de cartas que um jogador tem varia ao longo do jogo (o objetivo é descartar todas as cartas). Vamos assumir nesta questão que existe um limite máximo de 4 cartas que um jogador pode ter na mão. Neste caso, poderíamos usar uma estrutura para representar a mão do jogador (veremos nas próximas aulas como definir tipos de dados com elementos de tamanhos variáveis). Defina o tipo de dado `Mão` para representar as cartas da mão de um jogador (*convenção usada na disciplina: os nomes dos tipos de dados iniciam com letra maiúscula, mas na definição da estrutura usa-se letra minúscula*).

```
;; -----------------
;; TIPO MÃO:
;; -----------------
(define-struct mão (.....)
;; Um elemento do conjunto Mão é
;;   .............     onde
;;  ..............
```

Para pensar: Note que um jogador pode ter menos do que 4 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 de UNO) para representar que alguma posição pode estar livre.

Defina uma constante chamada `LIVRE` para descrever a carta que representa uma posição livre da mão. Defina 4 constantes representando mãos de jogadores, sendo que no mínimo duas mãos devem ter posições livres.

5. Construa a função `conta-cartas` que, dada uma mão, devolve o número de cartas de UNO que há na mão (lembre que posições ocupadas pela constante `LIVRE` não representam cartas de UNO).

6. Construa a função `define-jogada` que, dada uma mão e uma carta da mesa, nesta ordem, devolve uma carta da mão que pode ser jogada sobre esta carta da mesa (se houver) e uma mensagem, que pode ser `"Segue o jogo"`,`"UNO"` ou `"Ganhei"`, conforme o caso (se a mão só tinha uma carta e ela foi selecionada, o jogador ganhou, se tinha 2 cartas e uma foi selecionada, deve dizer UNO, caso contrário, a mensagem deve informar que o jogo segue). Para realizar a seleção da carta, deve-se considerar a ordem das cartas na mão: uma carta tem prioridade sobre cartas em posições subsequentes (**não considere** aquela restrição de somente poder jogar o `CuringaCompra4` se não houver na mão outra carta que possa ser jogada). Assuma que a mão passada como entrada para esta função não está vazia. Se o jogador não tiver carta para jogar, isso deve ser indicado pela escolha da constante LIVRE como carta a ser jogada (neste caso, a mensagem deve ser `"Segue o jogo"`).

   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 `Resultado` (usando a convenção descrita acima, o comando deve ser `(define-struct resultado ....)`. Pode escolher os nomes que quiser para os campos, mas o par deve conter a carta e a string, nesta ordem.

7. Faça a função `desenha-carta` que, dada uma carta, gera uma imagem para esta carta, inspirada nas cartas de um jogo de UNO. Se a entrada for a constante LIVRE, desenha apenas um retângulo escrito "LIVRE" dentro. *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*

   Dica: Para construir sua função, é útil fazer uma função auxiliar que dada uma carta, devolve sua cor, em inglês. Ou seja, as cartas das cores `"azul"`, `"verde"`, `"amarelo"`, `"vermelho"` ou `"preto"` devem retornar as cores `"blue"`, `"green"`, `"yellow"`, `"red"` ou `"black"`, respectivamente. Lembre que TODAS as funções devem ter documentação completa.

8. (Desafio - Ponto extra) Desenvolva a função `mostra-jogada` que, dada uma mão e uma carta, representando a carta da mesa, gera uma imagem mostrando a mão, a carta da mesa, a carta selecionada para jogar (que pode ser uma carta ou a constante LIVRE, caso não seja possível jogar nenhuma carta da mão), e a mensagem `Segue o jogo`, `UNO` ou `Ganhei`, de acordo com o caso. 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.

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