Haskell Assignment #1

The Game of 7 (Sedma)

Rules

Sedma is a 4-card trick-and-draw game played with a 32-card piquet deck. The game is played by four players A, B, C, and D in fixed partnerships, sitting crosswise (A plays with C, B plays with D). The suits are irrelevant for this game, and the ranks are not ordered in a hierarchy. Aces and tens have card-point values of 10 points each, while all other cards have no card-point value. Every player is dealt 4 cards. The remaining cards form a stock from which the players fill up their hands while it lasts.

Round

The game is played in rounds. In one round, each player plays one card, and the four played cards constitute a trick. The first card is played by a leader and other players play their cards in a clockwise order. The players are completely free in which card to play to the trick. The last player to play a card of the same rank as the first card wins the trick and becomes the leader for the next round. The sevens function as jokers, that is, they assume the rank of the first card in the trick. However, if the first card is a seven, it just represents a seven. At the end of each round, the winner collects the trick and places it in front of him. Finally, each player takes one card from the stock (if available) starting with the winner who takes his card first. The first round is led by player A.

Counting

The game ends when all the cards have been played. At the end of the game, each player counts the number of aces and tens in the tricks he or she had won. Each ace or ten counts for 10 points. There are additional 10 points for the winner of the last trick. The players in each team sum their scores. The winning team takes either 1, 2, or 3 points for a single game. The winning team takes 3 points, if the loosing team was not able to win a single trick. If the loosing team won one or more tricks but no trick was worth points (that is, the winning team has 90 points) the winning team takes 2 points. Otherwise, the winning team takes 1 point for the game.

Assignment

Your task is write a Haskell function which given a sequence of cards played during the game computes the winning team and its score (1, 2, or 3 points). You **must** use the following Haskell types to represent input and output. The cards are represented as follows, for example, the value `(Card Heart R7)` represents the 7 of heart.

```
data Suit = Heart | Diamond | Spade | Club deriving Show
data Rank = R7 | R8 | R9 | R10 | RJ | RQ | RK | RA deriving Show
data Card = Card Suit Rank deriving Show
type Cards = [Card]
```

The following types **must** be used to represent the outcome. For example, the value `(AC, Two)` represents a game won by the team of players A and C with the score of 2 points.

```
data Team = AC | BD deriving Show
data Points = One | Two | Three deriving Show
type Winner = (Team, Points)
```

Your task is to write a Haskell function

```
replay :: Cards -> Maybe Winner
```

which computes the winning team and its score. The input to function `replay` is the list of the cards in the order they were played during the game. That is, the first card was played by player A, the second by player B, etc. The fifth card, however, was played by the winner of the first trick (that is, not necessarily by player A). Your implementation must check that the input is a valid sequence of cards which constitute a possible game record. For an invalid input, your function must return `Nothing`.

You must use the above Haskell types as they are, that is, you are not allowed to adjust the definitions, **not even the "deriving" clause**. If you need to make some type an instance of some type class, **you may use** the `instance` construct. Download the types module file SedmaDatatypes.hs and start your program with the following line.

```
import SedmaDatatypes
```

Your solution will be tested by a Haskell 98 interpreter and the above Haskell types module will be provided automatically. Do not include the module in your submission. You are allowed to use the functions from Haskell Prelude and you must use Haskell 98 language with no extensions.