Poker hand evaluator using Rust

A program that deals two 2-card poker hands and a shared five card pool (the flop/turn/river for those who know these poker terms) according to the rules of Texas Holdem poker. The program will determine the winning hand and return it. The rules of Texas Holdem and the various hand rankings can be found at the links below (ignore all talk of incremental betting/passing/folding, we aren't considering that):

https://www.partypoker.com/en/how-to-play/texas-holdem
https://www.fgbradleys.com/et_poker.asp

When determining the strength of each player's hand, consider the two cards that player dealt, as well as the five cards present in the shared pool. The stronger hand will be returned as the winner.

Program Input

The input to the program will be the first 9 values in a permutation of the integers 1-52. This represents a shuffling of a standard deck of cards. The order of suits in an unshuffled deck are Clubs, Diamonds, Hearts, Spades. Within each suit, the ranks are ordered from Ace, 2-10, Jack, Queen, King. The table below shows all 52 cards and their corresponding integer values in a shuffling.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
Clubs	Ace	2	3	4	5	6	7	8	9	10	Jack	Queen	King
	14	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**
Diamonds	Ace	2	3	4	5	6	7	8	9	10	Jack	Queen	King
	27	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**
Hearts	Ace	2	3	4	5	6	7	8	9	10	Jack	Queen	King
	40	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**	**51**	**52**
Spades	Ace	2	3	4	5	6	7	8	9	10	Jack	Queen	King

Thus, an input sequence that started with the integers [38, 48, 11, 6, …] would represent Queen of Hearts, 9 of Spades, Jack of Clubs, 6 of Clubs, and so on. The program is not tested with broken or invalid input. Do not have to error-check this.

The program will accept this permutation as input and use it to deal two poker hands of two cards each in an alternating fashion. I.e., the first card goes to hand 1, the second card goes to hand 2, the third card goes to hand 1, fourth to hand 2. The remaining five cards will form the shared pool. Once dealt, the program will analyze each hand according to the rules from the websites above and decide a winner. When determining hand strength, each player considers their own cards as well as those in the shared pool. Effectively, each player has seven cards from which they will form the strongest possible hand.

Tie Breaking

According to the standard rules of poker, the ranks of the cards are used to decide the winner when both hands have the same strength. For example, if both hands are a flush, then the hand with the card of highest rank is declared the winner. If both hands have a pair, then the hand whose pair is higher wins. For example, a pair of Kings beats a pair of Sevens. If both hands have the same pair, i.e. each hand has a pair of threes, then the hand with the next highest card wins (called the kicker).
It is possible for two hands to remain tied even after all tie-breaking mechanisms based on rank are considered. An absurd example would be if the five cards in the shared pool formed a royal flush, in which case both players would have the exact same best hand (the royal flush). The program will not be tested on such inputs. Assume that all inputs will produce a clear and unambiguous winner. To put it in poker terms, don't have to worry about any input that would result in the players splitting the pot.

Rust Requirements

In a single Rust file called Poker.rs, write a function called deal that accepts an array of integers as an argument and returns the winning hand. In Rust, the typing of the input argument and the return value are very important. The usage of the deal function must be as follows:

```
let perm:[u32;9] = [1, 2, 3, 4, 5, 6, 7, 8, 9];
let winner:Vec<String> = deal(perm);
```

The deal function should accept an array of 9 unsigned integers, and the deal function should return a vector of five Strings. The strings should be formatted in the same way as all previous assignments. **No 3rd party libraries!**

The submission should not include a main() function. The tester will implement main(). Finally, the deal function should be declared as public using the pub keyword. The required signature for the deal function is below:

```
pub fn deal(perm:[u32;9])->Vec<String>
{
        // Code
}
```