In this card game, you have a deck of 40 cards.

1,2,3,4,5,6,7,J,Q,K in 4 suits, so that is 40 cards total.

The value of J is 11, Q is 12, and K is 13.

1. Randomly assign 4 cards to the player.
2. Randomly assign 4 cards from the remaining deck to the dealer.
3. Display what cards each has. **No need to display the suit. In fact, suits do not matter**.

e.g. Player: 6, 3, J, 4

Dealer: K, 5, 7, 2

**These cards will be checked in 3 different ways as follows:**

1. Determine who wins in terms of the total value of the cards. In the above example, the player is 24 and the dealer is 27 so the dealer wins.
2. Determine who wins comparing the value of **each** card - from the biggest to the smallest

e.g. Player sorted: K, 6, 4, 2

Dealer sorted: K, 7, 5, 1

K-K is a tie, but 7 beats 6. The dealer wins.

6. Finally, determine who wins in terms of the number of pairs.

e.g. if the cards are: J 3 J 3 then there are two pairs

e.g. if the cards are: 1 1 2 1 then there is only one pair

**First answer the following questions:**

**Q1. How would you store the 40 cards so that it is easy to randomly select cards for the players and make sure the same card is not selected more than once?**

**Draw the data structure with its contents. Explain why this is efficient.**

**Hint: srand (time(NULL)); // seed for random number generator #include<cstdlib> is needed**

**Int X = rand() % MAX; // generates a random number X between 0 and MAX**

**Q2. How would you store the 4 cards for each player so that the steps 4, 5 and 6 are easily done? Perhaps you do not want to sort/re-organize the cards for each step? Or, it does not matter?**

**Draw the data structure with example cards. Explain why this is efficient.**

**Implement your solution.**