The dealer's job in the card game Black Jack is rather simple (at least when it comes to the rules). The basic rule is that the dealer should take a card as long as the sum of all cards is below 17. The cards 2-10 has their respective face value while all face cards (jack, queen and king) are worth 10. Aces are worth 11 as long as the total sum is below 21, in that case the ace is worth 1 instead. In this task, we'll simplify the rules even more and will always count aces to be worth 1.

Your task is to use the standard library components to print a set of cards that the dealer could draw according to the following algorithm.

1. Create a vector<**int**>, `v` with 52 elements.
2. Fill `v` with the numbers 1..52. These numbers represent the cards in a deck of cards where the numbers 1..13 represent ace of spades to king of spades , 14..26 are ace to king of diamonds, 27..39 are clubs and 40..52 represent hearts.
3. Shuffle `v` randomly.
4. Create a new vector<**int**>, `card_values` without values.
5. For each value in `v`: convert it to the card value (range $[1, 13]$) and save the converted value in `card_values`. The value 52 from `v` should be transformed to 13 and stored in `card_values` while 28 would convert into 2.
6. Convert all face cards in `card_values` (card with value 11-13 after the last step) into 10.
7. Create a new vector<**int**>, `sums` with the same size as `v`.
8. Use an algorithm to make each element i in `sums` to be the accumulated sum of all elements from `card_values` with index $0..i$.
9. Find the index of the first element, `N`, in `sums` that has a value of at least 17.
10. Print all elements with index $0..N$ from `v` (printing as numbers 1..52 is fine, no need to convert into suit and value).