CS 454 – Node.js & Angular.js

Homework 2

Array Flattening – 20pts

Please submit this solution as a single file named: ***FlatArrays.js***

Write a function named **flattenArray** that uses the **reduce** method in combination with the **concat** method to "flatten" an array of arrays into a single array that has all the elements of the input arrays. This function should apply to *all* arrays. (Hint: use *Array.prototype* to define your function)

```
var arrays = [[1, 2, 3], [4, 5], [6]];
console.log( arrays.flattenArray )
// → [1, 2, 3, 4, 5, 6]
```

Your code should execute the code above.

removeSubstring Prototype – 30pts

Please submit this solution as a single file named: ***RemoveSubstring.js***

Implement a **removeSubstring** function that allows a specific sub-string to be removed from *any* string object, if the substring is present. Note, your function should take a second, optional numeric parameter. This parameter will be used in the event that the substring occurs more than once. If present, this parameter will indicate the number of times the substring should be removed from the string. If omitted, you should remove ALL occurrences of the substring.

```
var str1 = 'aaa';
var newStr1 = str1.removeSubstring('a', 2); // newStr = 'a'
console.log( newStr1 )

var str2 = 'aaabbbb';
var newStr2 = str2.removeSubstring('a'); // newStr = 'bbbb'
```

Your code should execute the code above.

Deck of Cards – 50pts

Please submit this solution as a single file named: ***Cards.js***

Create a "class" **card** that has the following immutable properties:

- *rank* : String
    - The rank of the card can be one of the following: Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King
- *suit : String*
    - *The suit of the card can be one of the following: Hearts, Clubs, Diamonds, Spades*

The **card** class should also contain a **.value()** method that returns the numeric value of the card. Note: an Ace is worth 1. Jacks, Queens, and Kings are worth 10, and all other cards assume the value of their corresponding rank.

Finally, you should override the **toString** method (Hint: use card.prototype to implement this). The **toString** method should print the rank and suit of the card in a human-readable format (see examples below).

Example usages of your card class are as follows:

```
var card1 = new card('Ace', 'Spades');  // Creates the Ace of Spades card
var card2 = new card('9', 'Hearts');       // Creates the 9 of Hearts card
console.log( card1.value() ); // Prints 11
console.log( card1 ); // Ace of Spades
console.log( card2.value() ); // Prints 9
console.log( card2 ); // 9 of Hearts
```

Next, create a "class" **deck** that acts as a container for 52 cards. That is to say, the **deck** should contain a private array named **cards** that is comprised of 52 card objects.
- Upon instantiating a deck, all 52 card objects should also be instantiated.

The **deck** class should have the following functions:

- **deal()**
    - The **deal()** function will remove and return the "top" card from the **cards** array. Assume the top card is at index zero (0) of the array.
- **remaining()**
    - The **remaining()** function returns the number of card objects remaining in the deck. Note, as you deal cards, the number of cards remaining will decrease.
- **shuffle()**
    - The **shuffle()** function will randomize the order of the cards currently in the deck.
- **reset()**
    - The **reset()** function will re-initialize the deck with all 52 cards.

Your code should execute the following:

```
var myDeck = new deck();
myDeck.shuffle();

var hand1 = [];
hand1.push( deck.deal() );
hand1.push( deck.deal() );

// Note, the output below will vary since the deck will be shuffled
console.log( ‘Hand 1: ‘, hand1[0], hand1[1] );  // Hand 1:  2 of Hearts  9 of
Clubs

var hand2 = [];
hand2.push( deck.deal() );
hand2.push( deck.deal() );

console.log( ‘Hand 2: ‘, hand2[0], hand2[1] );  // Hand 2: Queen of Diamonds
Jack of Clubs

console.log(‘The value of Hand 1 is: ‘, hand1[0].value() + hand1[1].value());
console.log(‘The value of Hand 2 is: ‘, hand2[0].value() + hand2[1].value());

console.log(‘There are ‘, deck.remaining(), ‘ cards remaining in the deck’);
//48
deck.reset();
console.log(‘There are ‘, deck.remaining(), ‘ cards remaining in the deck’);
//52
```