Stack - Practice Problems

Do the given problems using JavaScript.

1. Write a Program to keep track of the maximum element of the stack.

   Example:- Input: {50,70,80}

   Output: 50 70 80

2. Write a Program to reverse a stack with the help of another empty stack.
3. Write a Program to input an array and find out the next greater element for every element in the array.

   Example:- Input: [4 , 5 , 2 , 25]

   Output: 5 25 25 -1

4. Write a Program to return the smallest possible integer after removing k digits from num, where string num representing a non-negative integer num, and an integer k, is given as input.

   Example:- Input: num = "1432219", k = 3

   Output: "1219"

Solutions

1.

```
    let mainStack = [];

    let trackStack = [];

    function push(x)
    {
        mainStack.push(x);
        if (mainStack.length == 1)
        {
            trackStack.push(x);
            return;
        }

        // If current element is greater than
        // the top element of track stack, push
        // the current element to track stack
        // otherwise push the element at top of
        // track stack again into it.
        if (x > trackStack[trackStack.length - 1])
            trackStack.push(x);
        else
            trackStack.push(trackStack[trackStack.length - 1]);
    }

    function getMax()
    {
        return trackStack[trackStack.length - 1];
    }

    function pop()
    {
        mainStack.pop();
        trackStack.pop();
    }

    push(50);
    document.write(getMax() + "</br>");
    push(70);
    document.write(getMax() + "</br>");
    push(80);
    document.write(getMax());
```

2.

```
function transfer(s1, s2, n)
{
    for (i = 0; i < n; i++) {

        // Store the top element
        // in a temporary variable
        var temp = s1[s1.length-1];

        // Pop out of the stack
        s1.pop();

        // Push it into s2
        s2.push(temp);
    }
}

// Function to reverse a stack using another stack
function reverse_stack_by_using_extra_stack(s,n)
{
    var s2 = [];
    var i;
    for (i = 0; i < n; i++) {

        // Store the top element
        // of the given stack
        var x = s[s.length-1];

        // Pop that element
        // out of the stack
        s.pop();

        transfer(s, s2, n - i - 1);
        s.push(x);
        transfer(s2, s, n - i - 1);
    }
}
    var n = 5;
    var s = []
    s.push(1);
    s.push(2);
    s.push(3);
    s.push(4);
    s.push(5);

    reverse_stack_by_using_extra_stack(s, n);
    var i;
    for (i = 0; i < n; i++) {
        document.write(s[s.length-1] + ' ');
        s.pop();
    }
```

3.

```
function printNGE(arr, n)
      {
        var next, i, j;
        for (i = 0; i < n; i++)
        {
          next = -1;
          for (j = i + 1; j < n; j++)
          {
            if (arr[i] < arr[j])
            {
              next = arr[j];
              break;
            }
          }
          document.write(arr[i] + " -- " + next);
          document.write("<br>");
        }
      }
      var arr = [4,5,2,25];
      var n = arr.length;
      printNGE(arr, n);
```

4.

```
var removeKdigits = function(num, k) {
    const stack = [];
    let removed = 0;
    for(let n of num) {
        while(stack.length && n < stack[stack.length-1] && removed < k) {
            stack.pop();
            removed += 1;
        }
        stack.push(n);
    }

    // remove all remaining large numbers
    while(removed < k) {
        stack.pop();
        removed += 1;
    }

    // remove all beginning zeroes
    while(stack.length && stack[0] === '0') {
        stack.shift();
    }

    return stack.length ? stack.join('') : '0';
};
```