In [ ]:

Question
https://leetcode.com/problems/next-greater-element-ii/ (https://leetcode.com/problems/next-greater-element-ii/)

```
/*
n=4 01234567
01    01
1 5 3 4
0 1 2 3

KETAN
*/
class Solution {
    public int[] nextGreaterElements(int[] nums) {
        int n = nums.length;
        int[] res = new int[n];
        Arrays.fill(res, -1);
        Stack<Integer> stack = new Stack<>();
        for (int i = 0; i < n*2; i++) {
           while(!stack.isEmpty() && nums[stack.peek()] < nums[i % n]){
               res[stack.pop()] = nums[i%n];
           }
           if(i<n) stack.push(i);
        }
        return res;
    }
}
```

```
class Solution {
public:
    // 1 5 3 4
    // 5 -1 4 5

    vector<int> nextGreaterElements(vector<int>& nums) {
        //[1,2,3,4,3]

        //Solution - 2
        int curr_max = INT_MIN;
        int max_idx = 0;

        // Find the max element and index of the element
        for (int i=0; i<nums.size(); i++) {

            if (nums[i] > curr_max) {
                curr_max = nums[i];
                max_idx = i;
            }
        }

        vector<int> result(nums.size());
        result[max_idx] = -1;
        int curr_idx = max_idx-1;

        // Create the stack and push the max element initially
        stack<int> s;
        s.push(curr_max);

        //Iterator from the prev idx of max_idx to max_idx (circular)
        while (curr_idx != max_idx) {

            if (curr_idx == -1)
                curr_idx = nums.size()-1;

            while(!s.empty() && s.top() <= nums[curr_idx]) {
                s.pop();
            }

            if (s.empty()) {
                result[curr_idx] = -1;
            } else {
                result[curr_idx] = s.top();
            }

            s.push(nums[curr_idx]);
            curr_idx--;

            if (curr_idx == -1)
                curr_idx = nums.size()-1;
        }

        return result;
    }
};
```

In [ ]:

Question
https://leetcode.com/problems/power-of-two/ (https://leetcode.com/problems/power-of-two/)

```
TC: SC: log N
bool isPowerOfTwo(int n) {
    if(n==1) return true;
    else if(n==0) return false;
         return (n%2==0&&isPowerOfTwo(n/2));
}

// O(1)
class Solution {
    public boolean isPowerOfTwo(int n) {
        return n < 0 ? false : Integer.bitCount(n) == 1;
    }
}

// O(1)
// surendhar
class Solution {

    // 8        7
    // 1000  0111 => 0
    //
    // 10    9
    // 1010  1001 => 1000

    // 40       39
    // 101000   100111
    public boolean isPowerOfTwo(int n) {
        // negative or zero
        if (n <= 0){
            return false;
        }

        //
        else if ((n & (n - 1)) == 0){
            return true;
        }
        else {
            return false;
        }
    }
}


public static int bitCount(int i) {
        // HD, Figure 5-2
        i = i - ((i >>> 1) & 0x55555555);
        i = (i & 0x33333333) + ((i >>> 2) & 0x33333333);
        i = (i + (i >>> 4)) & 0x0f0f0f0f;
        i = i + (i >>> 8);
        i = i + (i >>> 16);
        return i & 0x3f;
}
```

In [ ]:

Question
https://leetcode.com/problems/decode-string/ (https://leetcode.com/problems/decode-string/)

In [ ]:

Searching

In [ ]:
```
Find a value in an array type
[10 7 2 3 4 5 6]
O(N)
Linear search

found ->  0 <= result < size of array
not found -> -1

PreCondition: Array is sorted
[1 3 4 6 7 8 9]
X

[1 3 4 7 8 9]
O(log N)
Binary Search

Divide and Conquer

int bSearch(std::vector<int> data, int value) {
    int start, end, mid;
    start = 0;
    end = data.size() - 1;

    while(start <= end) {
        mid = (start + end)/2;  // start + (end-start)/2

        if(data[mid] == value) return mid;

        if (value < data[mid]) {
            end = mid-1;
        } else {
            start = mid+1;
        }

    }
    return -1;
}
// TC: log N
// SC: O(1)

// value=2
// [1 3 4 6 7 8 9]
//  0 1 2 3 4 5 6
// start 0 0 0 1
// end   6 2 0 0
// mid   3 1 0

// 8
// [1 3 4 6 7 8 9]
//  0 1 2 3 4 5 6
// start 0 4
// end   6 6
// mid   3 5
```

In [ ]:

Question-1
https://leetcode.com/problems/binary-search/ (https://leetcode.com/problems/binary-search/)

In [ ]:

In [ ]:

Question-2
https://leetcode.com/problems/search-in-rotated-sorted-array/ (https://leetcode.com/problems/search-in-rotated-sorted-array/)

In [ ]:

In [ ]:

DIY:
https://leetcode.com/problems/find-peak-element/ (https://leetcode.com/problems/find-peak-element/)

In [ ]:

In [ ]:

In [ ]: