<u>Stack</u>:

1. Based on the concept of Last In First Out (LIFO).
2. Real-life examples of stacks: simple
3. Major operations in stacks:
    a. **Push**: Add an element to the stack if memory is available
    b. **Pop**: Remove the top element of the stack
    c. **Peek or Top**: Return the top element of the stack
    d. **size()**: returns the size of the stack
    e. isEmpty
4. Can be implemented using linked list or arrays:
    a. Arrays are preferred due to less dynamic involvement of memory allocation!
    b. LinkedList is preferred when the maximum possible size is not known
    c. Inbuilt stack functions/classes are safe to use!

Stack Problems:

Check If Parentheses are Balanced:

Given an expression string **x**. Examine whether the pairs and the orders of “{“,”}”,”(“,”)”,”[“,”]” are correct in exp.
For example, the function should return 'true' for exp = “[()]{}{[()()]()}” and 'false' for exp = “[(])”.

```
Input:
{([])}
Output:
true
Explanation:
{ ( [ ] ) }. Same colored brackets can form
balaced pairs, with 0 number of
unbalanced bracket.
```

```
Input:
([]
Output:
false
Explanation:
([]. Here square bracket is balanced but
the small bracket is not balanced and
Hence , the output will be unbalanced.
```

Next Greater Element

Given an array **arr[ ]** of size **N** having distinct elements, the task is to find the next greater element for each element of the array in order of their appearance in the array.
Next greater element of an element in the array is the nearest element on the right which is greater than the current element.
If there does not exist next greater of current element, then next greater element for current element is -1. For example, next greater of the last element is always -1.

```
Input:
N = 5, arr[] [6 8 0 1 3]
Output:
8 -1 1 3 -1
Explanation:
In the array, the next larger element to
6 is 8, for 8 there is no larger elements
hence it is -1, for 0 it is 1 , for 1 it
is 3 and then for 3 there is no larger
element on right and hence -1.
```

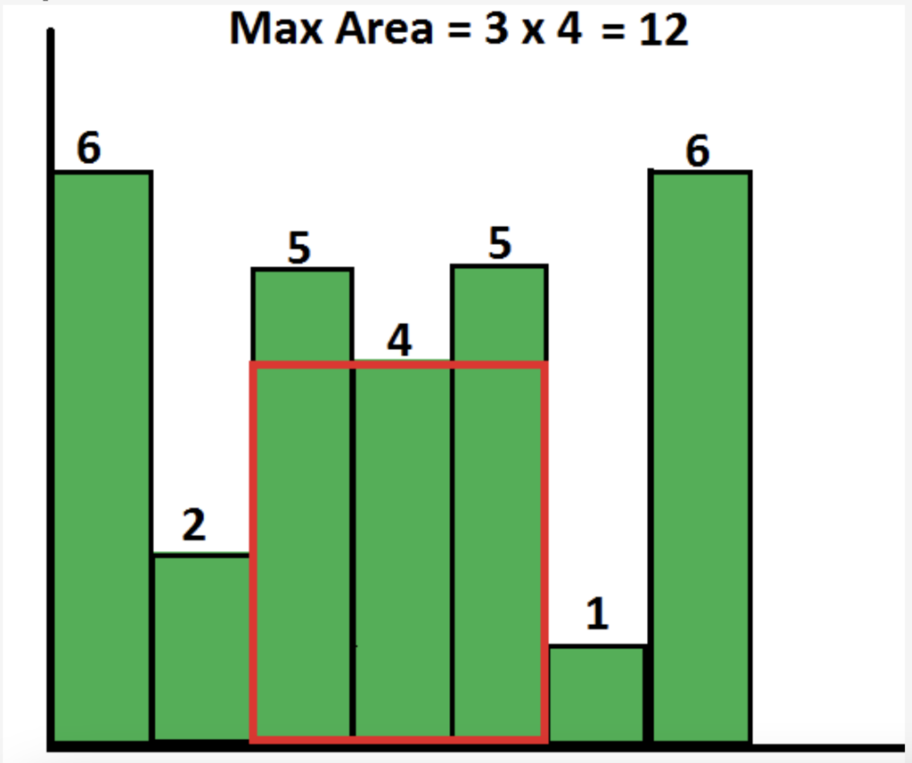
Ques-1: Can you now write an algorithm to find the immediate previous greater element for each element in the array?

Ques-2: Using the knowledge of finding immediate previous and next greater elements, can you write similar algorithms to find the immediate previous and next smaller elements for each element in the array?

If the answer to both of the above questions is "YES", try solving this:

Find the largest rectangular area possible in a given histogram where the largest rectangle can be made of a number of contiguous bars. For simplicity, assume that all bars have the same width and the width is **1 unit**, there will be **N** bars height of each bar will be given by the array **arr**.

```
N = 7
arr[] = {6,2,5,4,5,1,6}
Output: 12
Explanation:
```



<u>Queue</u>:

1. Based on the concept of First In First Out (FIFO)
2. Real-life examples of stacks: any queue
3. Major operations in queues:
    a. **Enqueue:** Adds an item to the queue. If the queue is full, then it is said to be an Overflow condition.
    b. **Dequeue:** Removes an item from the queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an Underflow condition.
    c. **Front:** Get the front item from the queue.
    d. **size()**: returns the size of the queue
    e. isEmpty

Queue Problems:

Implement the push and pop functionality of a Queue using two Stacks.

Generate Binary Numbers

Given a number **N**. The task is to generate and print all **binary numbers with decimal values** from **1 to N**.

Example 1:

```
Input:
N = 2
Output:
1 10
Explanation:
Binary numbers from
1 to 2 are 1 and 10.
```

Example 2:

```
Input:
N = 5
Output:
1 10 11 100 101
Explanation:
Binary numbers from
1 to 5 are 1 , 10 , 11 , 100 and 101.
```