Stack

Introduction to Stack

- Stack is a linear data structure that follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out).
- Mainly the following three basic operations are performed in the stack:
- **Push:** Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
- **Pop:** Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.
- **Peek or Top:** Returns the top element of the stack.
- **isEmpty:** Returns true if the stack is empty, else false.

Check for Balanced Parentheses

- **Problem Statement:** Check Balanced Parentheses. Given string str containing just the characters '(', ')', '{', '}', '[' and ']', check if the input string is valid and return true if the string is balanced otherwise return false.
- **Note**: string str is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.

```
Input: str = "( )[ { } ( ) ]"
Output: True
```

Next Greater Element Using Stack

- Given an array, print the Next Greater Element (NGE) for every element. The Next greater Element for an element x is the first greater element on the right side of x in the array. Elements for which no greater element exist, consider the next greater element as -1.

Example 1: Input: `N = 11, A[] = {3,10,4,2,1,2,6,1,7,2,9}`
Output: `10,-1,6,6,2,6,7,7,9,9,-1`

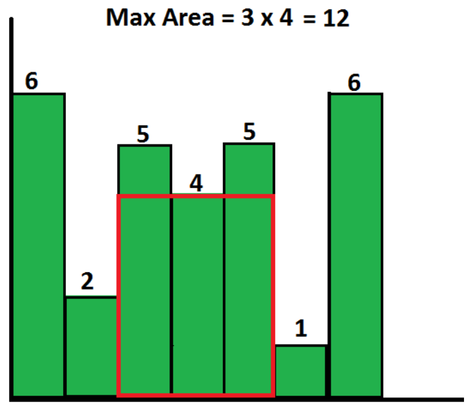
Example 2: Input: `N = 6, A[] = {5,7,1,7,6,0}`
Output: `7,-1,7,-1,0,-1`

Area of largest rectangle in Histogram

- **Problem Statement:** Given an array of integers heights representing the histogram's bar height where the width of each bar is 1 return the area of the largest rectangle in histogram.

```
Input: N =6, heights[] = {2,1,5,6,2,3}
Output: 10
```

{6, 2, 5, 4, 5, 1, 6}. Output:12



Online Stock Span

Design an algorithm that collects daily price quotes for some stock and returns **the span** of that stock's price for the current day.
The **span** of the stock's price today is defined as the maximum number of consecutive days (starting from today and going backward) for which the stock price was less than or equal to today's price.

- For example, if the price of a stock over the next `7` days were `[100,80,60,70,60,75,85]`, then the stock spans would be `[1,1,1,2,1,4,6]`.

Implement the `StockSpanner` class:

- `StockSpanner()` Initializes the object of the class.
- `int next(int price)` Returns the **span** of the stock's price given that today's price is `price`.

Input ["StockSpanner", "next", "next", "next", "next", "next", "next", "next"] [[], [100], [80], [60], [70], [60], [75], [85]]
Output [null, 1, 1, 1, 2, 1, 4, 6]