**StockSpanProblem**

**Q:** The stock span problem is a financial problem where we have a series of n daily price quotes for a stock and we need to calculate span of stock’s price for all n days.

The span Si of the stock’s price on a given day i is defined as the maximum number of consecutive days just before the given day, for which the price of the stock on the current day is less than or equal to its price on the given day.

**Test Cases:**

**Input:** [100, 80, 60, 70, 60, 75, 85]

**Output:** [1, 1, 1, 2, 1, 4, 6]

**Input:** [120, 50, 85, 20, 100]

**Output:** [1, 1, 2, 1, 4]

**Input:** [10, 20, 30, 40, 50, 60]

**Output:** [1, 2, 3, 4, 5, 6]

**Algorithm:**

1. Create an array span[ ] to keep the record of the span of the stock's price.
2. Insert 1 at the 0th index of span[ ] i.e. span[0]=1.
3. Push the index of the 1st stock in the stack.
4. Read the next stock value from the stock[ ] array
5. Pop elements from stack while stack is not empty and stock[s.peek( )] < stock[ i ].
6. If the stack gets empty than the price of stock will be greater than the price of stocks of all the previous days of the current stock otherwise price of the stock is greater than the elements after the top of the stack.

i.e. if stack become s empty, span[i] = i+1;

else span[i] = i-s.peek();

1. push the index value of the current stock to the stack.
2. Goto step 4.

**Example:**

Input stock[ ]=[10, 20, 30, 40]

* span[0]=1 and s.push(0) (Index of the first element is pushed to the stack.)
* Pop 0 form the stack as 10<20
* Stack gets empty so span for 20 becomes 2. (i+1)
* Push 2 to the stack.
* Pop 2 from the stack as 20<30.
* Stack gets empty again so, span for 30 becomes 3.
* Push 3 to the stack.
* Pop 3 from the stack as 30<40.
* Stack gets empty hence span for 40 is 4.
* Push it back to the stack.
* Print all the values of the span.