MINIMUM STACK

What is a minimum stack

A normal stack with an extra function to return the minimum element

Brute force solution is to just iterate through the stack everytime but that is O(N).

Better Solution is to make the stack store a pair of elements where the second element stores the current minimum element. Hence the top pair will always store the active minimum. But space complexity is O(2N) here.

Optimal Solution is to just store an extra minimum variable such as

		Stack	Min
push(12)		12	~~10~~ 12
push(15)		12 15	12
push(10)		12 15 8	~~12~~ 10
getMin	:	10	
pop		12 15	~~10~~ 12
getMin	:	12	
top	:	15	

Whenever you encounter an element lesser than minimum, dont store it in stack, store it in extra variable. Push

$$2 \times val - prevmin$$

to the stack instead.

When getMin comes, just return min element. When pop comes, check if top value in stack is lesser than min (i.e. a calculated one) or greater than min (i.e. a normal one). Eg: In above example when you pop 8, $8 < 10$ (hence its a calculated value).

Helps us recognize when was min value changed.

Pseudocode:

```
class minStack {
   stack <int> st;
   int m = INT_MAX;
   public:
      void push(int x) {
         if (st.empty()) {
            m = x;
            st.push(x);
         } else if (x < m) {
            st.push(2 * x - m);
```

```
            m = x;
        } else {
            st.push(x);
        }
    }
    void pop() {
        if (st.top() < m) {
            m = 2 * m - st.top();
            st.pop();
        } else {
            st.pop();
        }
    }
    int top() {
        if(st.top() < m) {
            return m;
        } else {
            return st.top();
        }
    }
    int getMin() {
        return m;
    }
}
```