**Assignment 6**

**Question 1**

**Write a function to find the maximum element in the stack.**

* **Method 1 (Brute-force):** We keep pushing the elements in the main stack and whenever we are asked to return the maximum element, we traverse the stack and print the max element.

Time Complexity: O(n)

Auxiliary Space: O(1)

* **Method 2 (Efficient):** An efficient approach would be to maintain an auxiliary stack while pushing element in the main stack. This auxiliary stack will keep track of the maximum element.

Below is the step by step algorithm to do this:

* Create an auxiliary stack, say ‘trackStack’ to keep the track of maximum element
* Push the first element to both mainStack and the trackStack.
* Now from the second element, push the element to the main stack. Compare the element with the top element of the track stack, if the current element is greater than the top of trackStack then push the current element to trackStack otherwise push the top element of trackStack again into it.
* If we pop an element from the main stack, then pop an element from the trackStack as well.
* Now to compute the maximum of the main stack at any point, we can simply print the top element of Track stack.

**Question 2**

**Write a function to find the minimum element in the stack.**

Get minimum element from stack in O(1).

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time (O(1)).

* push(x) — Push element x onto stack.
* pop() — Removes the element on top of the stack.
* top() — Get the top element.
* getMin() — Retrieve the minimum element in the stack.

We can solve this problem of min stack by using two stacks.

1. Declare two stacks. One is the main stack in which we push value as it is. In the second stack, we only push the minimum element present at that time.
2. Whenever we perform push operation in a stack.

* Push the value as it is in a first stack.
* For the second stack, push only the minimum value. For minimum value, compare the current value with the value present at the top of the stack.

Now, if peek() method is called on the second stack then we get the min element in O(1).