6.1. Add to the stack and queue the operation `getSum()` that returns the sum of the items in the stack/queue. Time complexity $O(1)$. Additional memory $O(1)$.

6.2. Add to the stack the operation `getMin()` that returns the smallest item on the stack. Time complexity $O(1)$. Additional memory $O(size)$ ($size$ is number of elements in the stack).

6.3. Using the stack, learn how to evaluate expressions in postfix notation (this is when the operator is placed after the arguments, for example, the expression `4 - ((1 + 2) * 3)` in postfix notation looks like this: `4 1 2 + 3 * -`.

6.4. Using the stack, learn how to check the correctness of the brackets sequence (this is the sequence of brackets of different types where each opening bracket corresponds to the closing bracket of the same type, like this: `[()](()[])`.

6.5. Given array of integers. For each i find maximal j such that $a[j] < a[i]$.

6.6. Let the allocation of a memory array of any size costs $O(1)$ time. Design vector with true (not amortized) cost of all operations $O(1)$ and memory $O(n)$. (Hint: you need to distribute the array copy operation over several operations).

6.7. Add to the queue the operation `getMin()` that returns the minimum item in the queue. Amortized time cost $O(1)$.

6.8. Implement the dequeue ousing three stacks with amortized time cost of all operations $O(1)$.