Jahongir Hayitov CS-01. 01.02.2023

Problem 1. In [Cormen, Section 16.1], a total amortised time complexity is computed for a sequence of n Increment operations starting from an initially zero counter. Compute the total amortised time complexity for a sequence of n Increment operations starting from a non-zero counter with value k. No justification is required, provide your answer using big-Oh notation.

```
1.  Increment(A, k) {
2.      i = 0;
3.      while(i<k && A[i] == 1) {
4.          A[i] = 0;
5.          i = i + 1;
6.      }
7.      if(i<k) {
8.          A[i] = 1;
9.      }
    }
```

ANSWER: O(N)

Problem 2. In [Cormen, Section 16.1], a stack with an extra operation Multipop is discussed. Provide an example of a sequence of Push, Pop, and Multipop operations on an initially empty stack, such that

- the actual total cost of the sequence is 5,
- the sequence contains one Pop, and one Multipop, and 3 Push operations, in some order,
- Multipop(k) must be used with $k \geq 2$.

No justification is required for this exercise.

```
1.  MULTIPOP(S, k) {
2.      while(! Stack-Empty(S) && k>0) {
3.           POP(S);
4.           k = k – 1;
5.      }
6.  }
```

ANSWER: PUSH(5) cost = 1, MULTIPOP(1) cost = 1, PUSH(3) cost = 1, PUSH(8) cost = 1, POP() cost = 1
Total cost = 5

Problem 3. Consider StackQueue, an implementation of the Queue ADT using a pair of stacks: a front stack
and a rear stack :

- A queue is empty when both stacks are empty.
- To perform offer(e), we push(e) into the rear stack.
- To perform poll(), we pop() from the front stack if it is not empty. If the front stack is empty, we repeatedly pop() elements from the rear stack and push them onto the front stack, until the rear stack is empty. Finally, we pop() from the front stack, since it is no longer empty.

Perform amortised time complexity analysis for a sequence of offer(e) and poll() operations performed on an initially empty StackQueue. You must apply either the accounting method or the potential method.

Assume that the execution cost (time) of push(e), pop(), isEmpty() for the underlying stack implementation is 1

ANSWER:

Since the Queue consists of a pair of Stack,
Actual cost per operation: offer(e) depends on push(e) which is 1 coin, and poll() depends on both of stacks and, we can substitute it with pop(), costs 1 coin, to take one element from the front, or k times pop() and push(e) to the front, costs 2k coins, a method to make the rear empty and push each to the front.
Invariant: Every element in the rear stack has 3 coins
Accounting($\check{c}_i$):
push(e) to the rear: assign 4 coins ($\check{c}_i$=4)

- 1 coin is for push(e) itself
- 2 coins are for pop elements from the rear stack and push them onto the front stack
- the other 1 coin is for pop() from the front

isEmpty(): assign 1 coin ($\check{c}_i$=1)
The worst case: when the queue is empty and we repeatedly call offer(e) and poll() methods one by one

to check the front is whether empty or not = 1 coin
to check the rear is whether empty or not = 1 coin
push to the rear = 4 coins
Total = 6 coins

Therefore, it costs 6n coins for sequence of n operations
Total amortized cost is **O(6n) = O(n) ▲**