Problem 4: [15 marks]

Implement a queue using two stacks S_1 and S_2 and and constant number of variables. Think of a suitable algorithm such that time complexity of N operations on the queue is O(N). (a) Explain the design of your algorithm and give pseudo-codes for the enqueue and dequeue functions. (b) Prove using amortized analysis that your queue takes O(1) time for enqueue and dequeue operations.

The method will be for ~~first~~ all no. of enqueues, we simply push the elements in stack -s1. After we encounter a pop, empty the whole s1 into s2. In this process the stack will be reversed. Now the element 1st enqueued is at top of S2. So simply pop from s2 & return if we encounter another pop, first check is s2 is empty or not. If it is not then just pop. else again empty s1 in S2 & pop. ~~Pushin~~

```
Enqueue( int a) {
    S1 → push(a)
}
```

```
dequeue () {
    if s2.isempty()
        while (!s1.isempty ()) {
            s2 → push(s1 → pop())
    if s2.isempty()
        raise exception
    return s2 → pop()
```

Amortized Time Comp.

① 'n' Pushes $\Rightarrow^{Push} = \frac{\sum^{n} O(1)}{n} = O(1)$

② 'n' pops :- As each element's presence in queue can have only operaⁿ/element, as it is first push, ~~copied~~ popped and pushed to s2 & poped from s2.

∴ Each element undergoes exactly 4 operaⁿ ⇒

For n element pushed & popped in any order ⇒

For n operaⁿ = $\boxed{O(4n) = O(n)}$ & $\boxed{Avg. = O(1) = Amortized}$