Makori Chacha 301414318

In this implementation, a queue is simulated using two stacks: inStack and outStack. The enqueue operation simply pushes the new element onto inStack, which takes constant time **O(1)**. The dequeue operation checks if outStack is empty. If it is, all elements from inStack are moved to outStack, which takes **O(n)** time in the worst case. However, because this shifting happens only when outStack is empty, the **amortized** time complexity for dequeue becomes **O(1)** over many operations. The front() operation behaves similarly.

When compared to the standard std::queue from the C++ Standard Library, which is usually implemented using std::deque, the standard queue provides **O(1)** time for both enqueue and dequeue operations in all cases, not just amortized. This makes it more efficient under heavy use.

Conclusion:

While the two-stack method is a clever workaround, the standard std::queue is better for systems like a self-serve coffee kiosk, where consistent and fast response time is important. The standard queue provides reliable constant time operations with no worst-case delays.