```
bool emptyStack( S ) {
    return ( emptyQueue(A) );
}
// Running time of emptyStack() : O(1)

void push( S, x ) {
    enqueue(A, x);
}
// Running time of push() : O(1)

Element pop ( S ) {
    Element element;
    while ( ! emptyQueue(A) ) {
        element = dequeue(A);
        if ( ! emptyQueue(A) )
            enqueue( B, element );
    }
    while ( ! emptyQueue(B) )
        enqueue( A, dequeue(B) );
    return element;
}
// Running time of pop() : O(n)
```

CS 251 Homework #4 Solution: *Stacks and Queues*

---

(1) Show how to implement a queue using two stacks. Specially, give the pseudo-code for enqueue(Q,x), dequeue(Q), initQueue(Q), and emptyQueue(Q) using only the initStack(S), emptyStack(S), push(S, x), and pop(S) stack operations.
Analyze the running time of the queue operations(i.e., what is their big-O).

```
void initQueue(Q) {
    initStack(A);
    initStack(B);
}
// Running  time of initQueue() : O(1)

bool emptyQueue() {
    return emptyStack(A);
}
// Running  time of emptyQueue() : O(1)

void enqueue(Q, x) {
    push(A, x);
}
// Running time of enqueue() : O(1)

Element Dequeue(Q) {
    while ( ! emptyStack(A) ) {
        push( B, pop(A) );
    }
    Element element = pop(B);
    while ( ! emptyStack(B) ) {
        push( A, pop(B));
    }
    return element;
}
// Running time of dequeue() : O(n)
```

---

(2) Show how to implement a stack using two queues. This is the converse of the problem above.
Analyze the running time of the stack operation.

```
void initStack (S) {
    initQueue(A);
    inutQueue(B);
}
// Running time of initStack() : O(1)
```