Homework 3

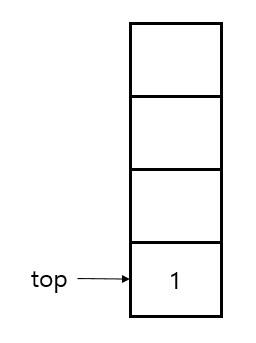
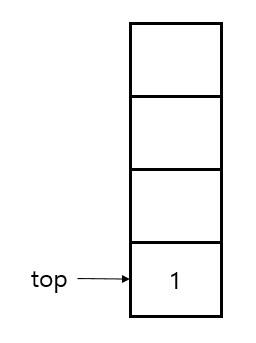
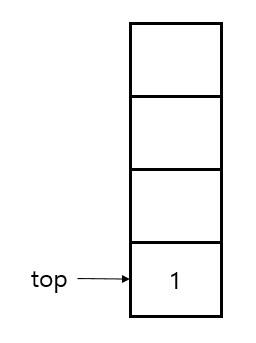
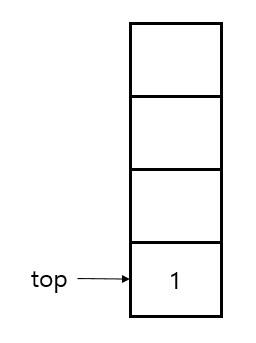
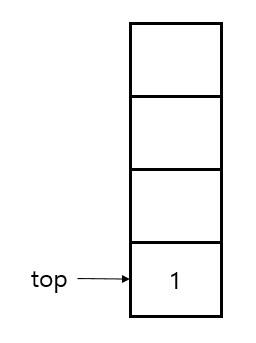
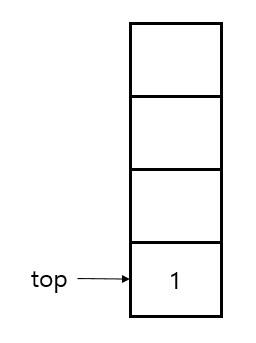
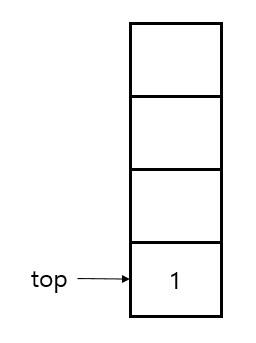
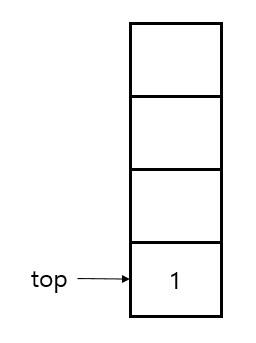
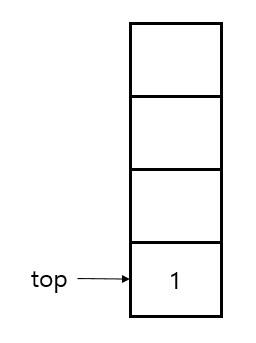
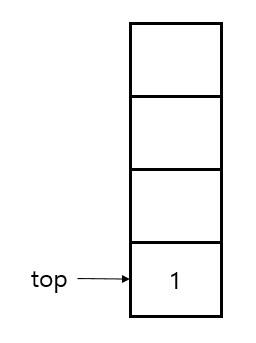
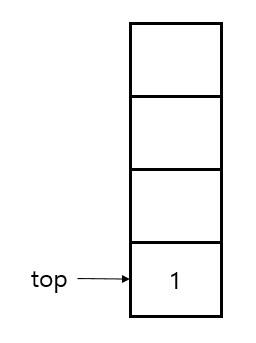
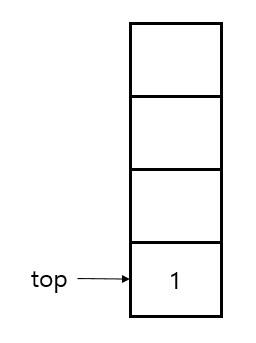
M1522.000900 Data Structure (2019 Fall)

2013-12815 Dongjoo Lee

**Question 1.**

1. **False.** Singly linked list has no member indicating the previous node. It takes 𝜃(n) time to  
    find previous node from the head.
2. **True.** Doubly linked list has previous node member so it can access to previous node in 𝜃(1)  
    no matter where the current position is.
3. ­**False.** For example, to insert new node, it is needed to access every next nodes in array-based  
    list. So, it takes 𝜃(n) in general.
4. **True.** Compared with Array-based list, that is required only memory for the information, It is  
    needed some extra memory space for linked list.
5. **True.** For example, when top of the stack points to the leftmost index, we can implement   
    “PUSH” by “return listArray[Top++];”.

**Question 2.**



4

5

5

2

5

5

2

10

2

5

5

**Question 3. (\* Let’s pretend there was Stack implements Stack ADT on page 6)**

**1st TODO:**

s.push(**Integer**.*toString*(i)); *// following left parenthesis’ position*

**2nd TODO:**

**if** (c **!=** ')') **continue**; *// If input sentence contains not only parenthesis*  
  
**if** (s.length() **==** 0) { *// there was no ( before* **return** i;  
}  
  
s.pop(); *// Pairing SUCCESS*  
**if** (s.length() **==** 0) { *// Input sentence is well balanced until now.*  
 position **= -**1;  
}

**Question 4.**

*/\*\* Put "it" in queue \*/***public void** enqueue(E it) {  
 *// Todo: implement enqueue using the given stacks* **this**.stack2.clear(); *// clear before use* **this**.stack2.push(it);  
  
 *// Main(old) -> Sub(empty)* **while** (**this**.stack1.length() **>** 0) {  
 **this**.stack2.push(**this**.stack1.pop());  
 }  
  
 *// Sub(Main+new item) -> Main(empty)* **this**.stack1.clear(); *// clear before use* **while** (**this**.stack2.length() **>** 0) {  
 **this**.stack1.push(**this**.stack2.pop());  
 }  
}

*/\*\* Remove and return front value \*/***public** E dequeue() {  
 *//Todo: implement dequeue using the given stacks* **this**.stack2.clear(); *// clear before use  
  
 // Main(old) -> Sub(empty)* **while** (**this**.stack1.length() **>** 0) {  
 **this**.stack2.push(**this**.stack1.pop());  
 }  
  
 E byeItem **= this**.stack2.pop(); *// item to be return  
  
 // Sub(Main-byeItem) -> Main(empty)* **this**.stack1.clear(); *// clear before use* **while** (**this**.stack2.length() **>** 0) {  
 **this**.stack1.push(**this**.stack2.pop());  
 }  
  
 **return** byeItem;  
}