Seoul National University

M1522.000900 Data Structure

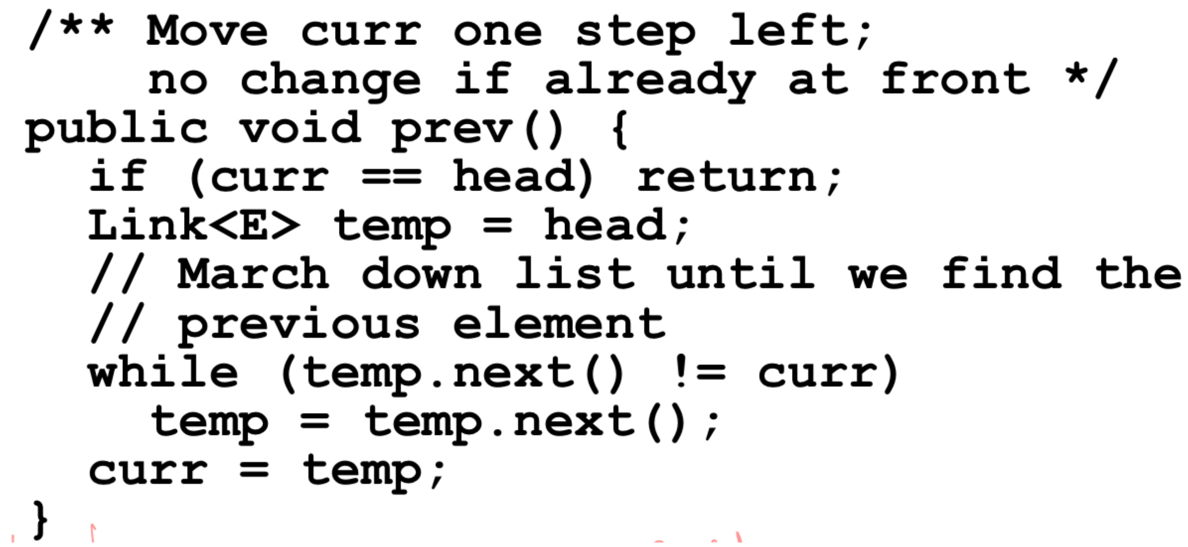
Homework 3: List, Stacks, and Queue (Chapter 4)

Computer Science & Engineering

2017-18538 Hwang Sun Young

Question 1

(1) False / We need to identify next of temp is curr. So the time complexity is θ(n).



```
/** Move curr one step left;
    no change if already at front */
public void prev() {
  if (curr == head) return;
  Link<E> temp = head;
  // March down list until we find the
  // previous element
  while (temp.next() != curr)
    temp = temp.next();
  curr = temp;
}
```

(2) True / prev pointer make us access previous at once.

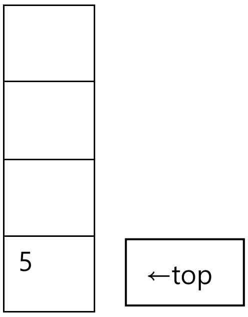
(3) False / In case of insertion, we need to move other element. Then we can insert new element to current position. In case of deletion we need to move other element after delete target element. This process cost θ(n).

(4) True / Unlike array-based list, Linked list needs additional space for pointer.

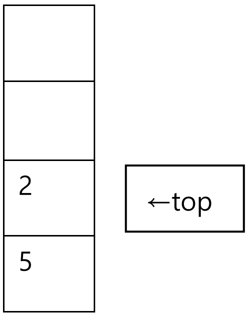
(5) False / if the top of the stack points to the leftmost index, the time complexity of "PUSH is θ(n) because it needs to shift element. But in the case of the rightmost index, the time complexity of "PUSH" is θ(1) because it simply needs to push an element.

Question 2

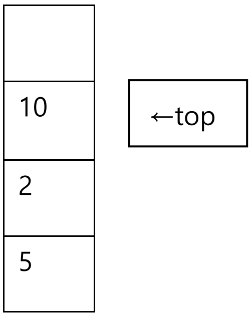
S.push(5);



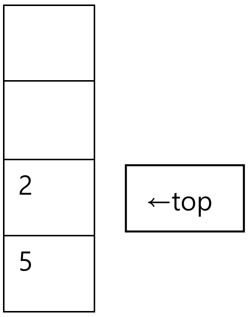
S.push(2);



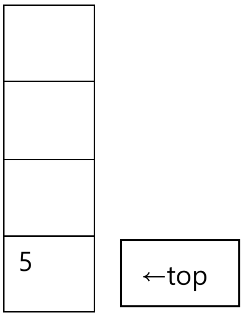
S.push(10);



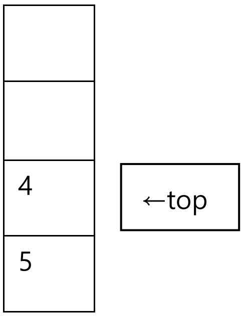
S.pop();



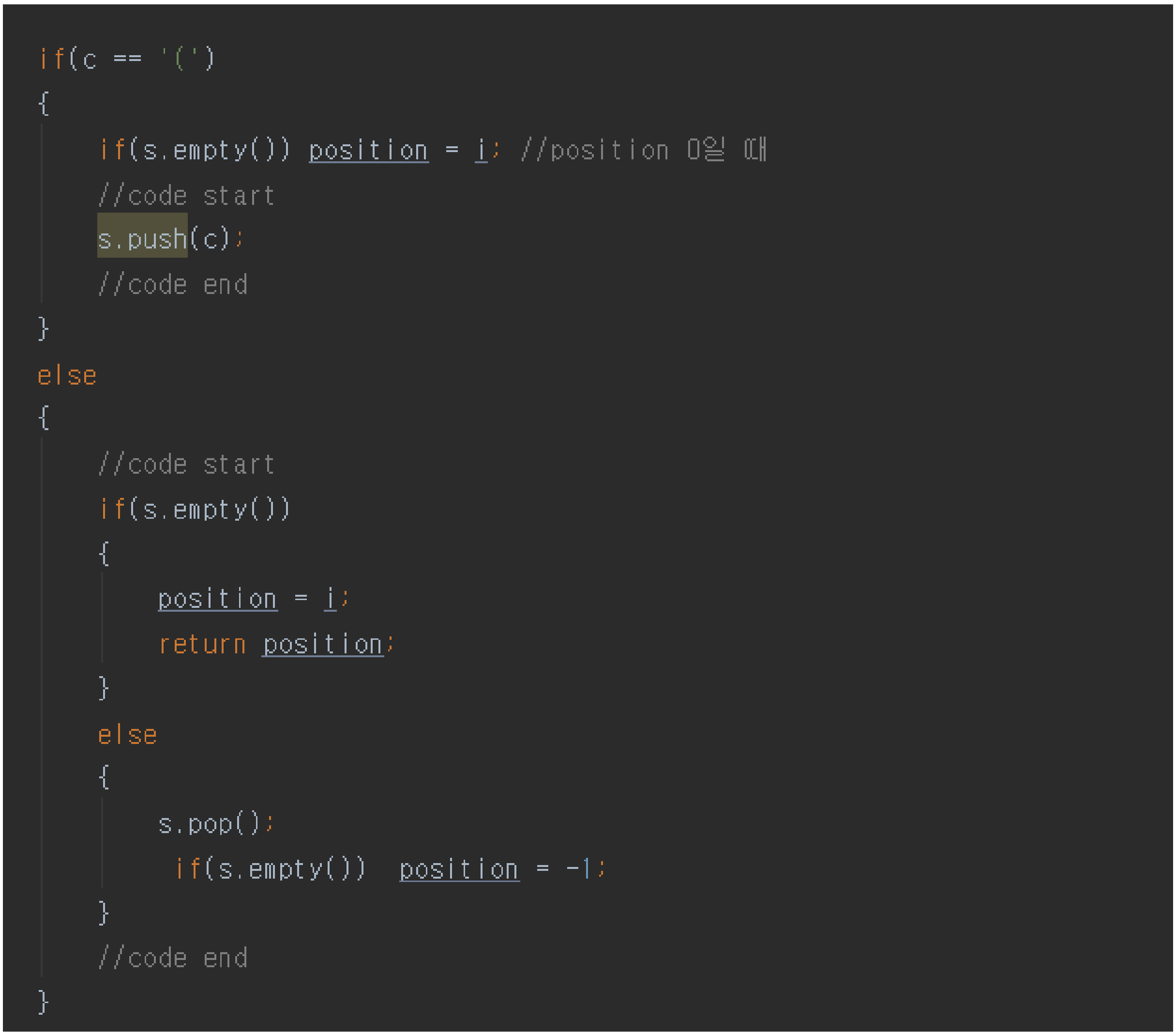
S.pop();



S.push(4);



Question 3



```
if(c == '(')
{
    if(s.empty()) position = i; //position 0일 때
    //code start
    s.push(c);
    //code end
}
else
{
    //code start
    if(s.empty())
    {
        position = i;
        return position;
    }
    else
    {
        s.pop();
         if(s.empty())  position = -1;
    }
    //code end
}
```

Question 4

```
public void enqueue(E it) {

            stack1.push(it);

}



public E dequeue() {

            if(stack2.length() == 0)

            {

                        while(!stack1.length() == 0)

                                    stack2.push(stack1.pop());
```

```
        }

        return stack2.pop();

}
```