**CS 3410: Data Structures**

**Homework 3 (Fall 2017)**

**Due date: Wednesday, September 27, 2017 by 2:00 P.M.**

**Rick Boles**

1. Suggest an implementation of a “sorted stack”. That is, write modified *push()* method so that the elements are always ordered (the largest item appearing at ‘top’). Without loss of generality, you can assume integer elements. Also you can assume the support of *pop(), peek(), isEmpty()*  and *isFull()* methods. However, you are not allowed to use anything other than stacks. Indicate the big-O complexity of your *push()* methods. [7 points]
2. **public** **static** **void** sortedMaxPush( **int** i){ **//O(n^2)**
3. Stack temp = **new** Stack();
4. Stack temp2 = **new** Stack();
5. **boolean** running= **true**;
6. **boolean** complete = **false**;
7. **if**(*s*.isEmpty())
8. *s*.push(i);
9. **else**{
10. **while**(!complete){
11. **while**(!*s*.isEmpty()){
12. **if**((**int**)*s*.peek() > i)
13. temp.push(*s*.pop());
14. **else** **if**((**int**)*s*.peek() <= i && running){
15. temp.push(i);
16. running = **false**;
17. }
18. **else** {
19. temp.push(*s*.pop());
21. }
23. }
25. **while**(!temp.isEmpty()){
26. *s*.push(temp.pop());
28. }
29. complete = **true**;
30. }
31. }

34. }
35. }
36. Is it possible to implement a simple Stack data structure using simple Queue (one or more) structure? If no, explain why it is not possible. If yes, suggest how the two stack operations (*push()*  and *pop()*) would be implemented using standard queue operations (enqueue() and dequeue()). Also indicate whether the big-oh time for *push()* and *pop()* would change. If they change, explain the new runtime complexities of the operations. [7 points]

It is possible with similar code below: (Push changes from O(1) to O(n) because the entire queue must be rebuilt each time it is called. Pop remains the same at O(1) since it only runs once)

Queue push(Queue q, int i){

Queue temp = new Queue();

temp.enqueue(i);

while(!q.isEmpty()){

temp.enqueue(q.dequeue());

}

}

int pop(Queue q){

return q.dequeue();

}

1. Suppose you have a stack S containing N distinct elements and a simple queue Q that is initially empty. Describe how you can use Q to scan S to see if it contains a certain element x, with the additional constraint that your algorithm must not violate the integrity of the stack (that is, the elements should remain in their original order after the scan is over). You may not use an extra array or other storage. Only S, Q, and a constant number of reference variables are allowed. What is the runtime complexity of your algorithm? [6 points]

So for this problem I would simply pop items out of the stack until I met the condition I was looking for with a loop and store the popped values within the que using the standard insert.

I would then remove all of the items from the queue using dequeue and push them back to the stack. This will leave the stack in reverse order.

I will then run a loop to take the items back out of stack and insert them back into the queue.

I would remove them from the queue one last time and put them back into the stack but this time the order will be correct.

I would have used the method from #2 but that would have given me a O(n^2). The way I have done it here is 4 loops but that keeps the complexity at O(n) because it runs through every value of stack or queue every loop used.

**Submission guideline:** You have the following options to submit your assignment. Choose only one option that suits you best.

1. Submit a handwritten copy in class. Writing should be clear (readable). Include **Course number** (CS3410), **Semester** (Fall 2017), **Assignment number** (Homework 3) and **your name** at the top. Failure to include these will not be accepted.
2. Scan your handwritten copy (including information mentioned in (i)) and submit it through BlazeVIEW dropbox. Scan resolution should be good (readable).
3. Type your answers on a document file using MS-Word. Include all information (mentioned in (i)) and submit through BlazeVIEW dropbox. Make sure symbols are appearing appropriately.