ECE608 chapter 10 problems
1) 10.1-2
2) 10.1-4
3) 10.2-4
4) 10.2-5
5) 10.4-4
function Push(S, i, x)
    if (S.head1 == S.head2-1) then
        "Overflow"
        return
    else
        if i==1 then
            S.head1 = S.head1+1
            S[S.head1] = x
        else
            S.head2 = S.head2-1
            S[S.head2] = x
        end if
    end if
end function

Figure 1: Push function taking the stack number and the value as inputs

(10.1-2) Have two heads for the two stacks given by S.head1 and S.head2. Initialize S.head1 = 0 and S.head2 = n+1. The push and pop operations are shown in Fig. 1, 2.

(10.1-4) Underflow happens while dequeueing an empty queue. To avoid this, check if Queue.head = Queue.tail and flag an error if this is true before trying to dequeue.
Overflow happens while trying to enqueue a full queue. To avoid this, check if Q.head = Queue.tail+1 and flag an error if this is true before enqueueing.

(10.2-4) The pseudo-code is shown in Fig. 3. The main idea is to make the value stored in the sentinel equal to the value being searched for. That way the loop terminates and it is also possible to determine if the search term is actually present in the list or not.

(10.2-5) Implementing a dictionary is very similar to storing a list of numbers. INSERT can be implemented by inserting at the head of the list in O(1) time. The other operations of DELETE and SEARCH will take O(n) in the worst case. This is because both operations may require going through all the elements present in the list.

(10.4-4) The pseudo-code for this is shown in Fig. 4
**function** Pop(S, i)
```
if (i == 1 AND S.head1 == 0) OR (i == 2 AND S.head2 == n+1) then
    "Underflow"
    return NaN
else
    if i == 1 then
        Value = S[S.head1]
        S.head1 = S.head1-1
        return Value
    else
        Value = S[S.head2]
        S.head2 = S.head2+1
        return Value
end if
end if
```
**end function**

Figure 2: Pop function taking the stack number as input

**function** SEARCH(head, searchTerm, sentinel)
```
sentinel.value = searchTerm
ptr = head
Found = 0
while (head != NIL AND Found == 0) do
    if (head.value == searchTerm) then
        Found = 1
        ptr = head
    end if
    head = head.next
end while
if (head == sentinel) then
    Found = 0
    ptr = NIL
end if
return ptr
```
**end function**

Figure 3: Modified search in a linked list
function PRINTNODE(T)
    if (T == NIL) then
        return
    else
        PRINTNODE(T.left)
        print "T.value"
        PRINTNODE(T.sibling)
    end if
end function

Figure 4: Print a left-child, right-sibling tree representation