# Part A: Corrections from Quiz 1

4. 
   - I got this question wrong originally as I was under the impression that the items would be pushed onto the back of the array, and thus I thought that the only pointer required was the one that pointed to the end. Of course, I now realize this is wrong as there would be no easy way to figure out the new "back" of the array after popping a value off the stack, as the linked list only has a pointer to the next element; being a singly-linked list, there does not exist a pointer to the previous element in the list.
   - The correct answer is to only have one pointer pointing to the front of the linked list. This allows the `push(char x)` and `pop()` operations to run in $\mathcal{O}(1)$ time, as both operations can be completed simply by manipulating the pointer at the front of the list (as well as the element to push/pop), without regards to the size of the linked list.
6. 
   - I originally believed that a linked list would be the best way to implement a stack, as each push/pop operation runs in $\mathcal{O}(1)$ time regardless of size, whereas an array would have to resize itself every so often in order to accommodate the new items in the list, which is not the most efficient operation.
   - My above notion is incorrect because in reality, all three methods would be effective to implement a stack, depending on its application. For instance, an array often requires less space to store than a linked list, as each element in a linked list must store a pointer to another element as well. The circular array has a similar concept. On the other hand, a linked list will never have to re-size itself, making it useful for very large lists of numbers.

# Part B: Debugging Q7 from the Quiz

1. The function incorrectly prints out that the sequence is balanced when the sequence contains more '+' than '-'. This is because the function does not check if the stack is empty before printing out that the sequence is balanced; if there are '+' still left in the stack after the sequence has been iterated through, the function should print out "unbalanced", as this means that the number of '+' is *not* equal to the number of '-'. For instance, the sequence "+ + −" is unbalanced, but will be incorrectly recognized as balanced by the function.
2. The correct implementation of the function (in pseudocode) is as follows:

```
String foo(char* sequence) {
  i = 0;
  declare a character stack;
  while (sequence[i] is not null) {
    if (character at sequence[i] is a '+')
      push it on the stack
    else {
      if (character at sequence[i] is a '−'
          and the stack contains at least one element)
        pop a character off the stack
      else
        print "unbalanced" and exit
    }
    i++;
  }
  if (stack is empty)
    print "balanced" and exit
  else
    print "unbalanced" and exit
}
```

   This code fixes the aforementioned bug by adding an if-statement at the end to check if the list is empty. The function will produce true if and only if the stack is empty as well, as this means that there must be exactly as many '+' as '-'.