CS 231: Fundamentals of Algorithms **Fall 2021**

Assignment #1

Your Name:

Collaborators:

Total time spent:

Problem 1. [15 points]
Consider the searching problem:
Input: An array of n numbers $A =< a_1, a_2, ..., a_n >$ and a value v.
Output: An index i such that $v = A[i]$, or the special value NIL if v does not appear in A.

1. Write pseudocode for the linear search, which scans through the array, looking for v.
2. What is the running time complexity of the algorithm you defined?
3. Prove that your algorithm is correct. Remember, your proof should be correct, clear, and concise.

Problem 2. [15 points]
In the following table, show the worst-case running times for the best algorithms for implementing the given operations for the dynamic data structure representations listed in the leftmost column. Please note the following:

- An argument of (S, key) implies we are passing a value of the object, and an argument of (S, obj) implies we are passing a pointer (or an index) to the object.
- The procedure Search(S,key) returns a pointer to the object matching the key, if found, and NIL otherwise.
- The procedure Insert(S,key) adds a new key to the beginning of the data structure.
- The procedure Delete(S,obj) deletes an object, specified by the pointer, from the data structure.

Data Structure	**Search(S,key)**	**Insert(S,key)**	**Delete(S,obj)**
Unsorted Array			
Sorted Array			
Unsorted Linked List			
Sorted Linked List			
Unsorted Doubly Linked List			
Sorted Doubly Linked List			

Problem 3. [10 points]
Write an algorithm that takes as input a sorted array, and returns the array after removing all duplicates. For example, an input array of $A = \{1, 1, 2, 3, 4, 4, 4, 7, 11\}$ should be converted into $A = \{1, 2, 3, 4, 7, 11, nil, nil, nil\}$.
Note: A nil value is equivalent to the null value in object oriented programming languages.

To get full credit, write an algorithm with $O(n)$ worst case running time complexity, without using any significant extra storage. Less efficient solutions are also accepted, but will get 80% of the grade.

Problem 4. [10 points]
Write an algorithm that takes as input two sorted singly-linked lists, and returns the merge of the two lists. The only part of the linked list that you are allowed to change is the forwarding links between the nodes.

The complexity of your algorithm must be $O(n+m)$, without using significant extra storage, in which $|A| = n$ and $|B| = m$.