

Northern Michigan University (Marquette Co, MI)

CS422-01-25W: Algorithms (Andrew A. Poe)
Practice Midterm Examination (Exam 1)

Name: _____
Monday 24 February 2025 9:00 A.M. EST

Time: 50 minutes

1. Given the following classes:

```
public class LL {  
    public Node head;  
}
```

```
public class Node {  
    public String s;  
    public Node next;  
}
```

Write the code for the method `public String SecondLast () {...}` in the LL class. This method returns the second-to-the-last string in alphabetical order. For example, if the linked list were: `I-->ATE-->A-->SANDWICH`, this method would return "I". Be sure you handle all special cases in a reasonable way. In particular, if there are two nodes with identical strings and that is the last string in alphabetical order, then that is the string you return since there is a "tie" for last.

You may assume that you have reasonable constructors and sets and gets. You are allowed to use helper methods, but you must write them if you do. Don't use loops; use recursion only. And make no modifications to the linked list or to any of its nodes.

2. Write the code for the following method: `public int DelDuplicates (String[] A) {...}`. The parameter A is an array of string that is **ALREADY KNOWN TO BE SORTED**. However, it may have duplicates. You are to remove the duplicates with in A. For example, if A is a six-element array: `{APPLE , APPLE , BOB , CAT , CAT , CAT}`. You are to reconfigure the array so that it no longer has duplicates: `{APPLE , BOB, CAT , ?? , ?? , ??}` (Since the array is of size 6 and there are only three distinct entries, I don't care what the final three entries are.) You are to return the number of distinct entries, in this case, 3. You can use helper methods but write them if you do. For this one, you can use loops or recursion or both; however, you can't use a second array. Move everything around in A.

Northern Michigan University (Marquette Co, MI)

CS422-01-25W: Algorithms (Andrew A. Poe)
Practice Midterm Examination (Exam 1)

Name: _____
Monday 24 February 2025 9:00 A.M. EST

3. I have a heap that looks like this: G F C D E A B. Pop the heap. What element gets popped off? What does the heap look like afterward? Show all steps.

4. Imagine I have an array that is already sorted (it is, but I don't know that it is). Consider the sorts we have discussed in class: Merge, Shell, Tree, Quick, Heap, Insertion. Which of these sorts runs considerably quicker if the array is already sorted. Which run marginally or “sort of” quicker? Which run about the same? Are there any that run SLOWER? Explain your answer; don't just make a table!![