Northern Michigan University (Marquette Co, MI)

CS422-01-25W: Algorithms (Andrew A. Poe) Name: _____
Midterm Examination (Exam 1) Friday 28 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 LL method `void DelLast (String s) { ... }`. This method removes the node containing s from the linked list, if such a node exists. If s appears more than once in the linked list, it deletes the node CLOSEST TO THE TAIL containing that string and ONLY that node. For example, if the list were A->B->C->B->D->B->E, and DelLast ("B") were called, the list would become A->B->C->B->D->E. You may assume reasonable constructors, sets, and gets already exist. You may use helper methods in either class, but you must write them if you do. Do not use loops; use recursion only. Do not create nodes or modify the data fields of existing nodes; use pointer manipulation only.

LL:

```
public void DelLast (String s) {

 if (head != null) {
  boolean DONE = head.DelLast ();
  if (!DONE && head.gets().compareTo(s)==0)
   head = head.getnext();
 }
}
```

Node:

```
public boolean DelLast (String s) {

 boolean DONE = false;
 if (next != null) {
  DONE = next.DelLast (s);
  if (!DONE && next.gets().compareTo(s)==0) {
   next = next.getnext();
   DONE = true;
  }
 }
 return DONE;
}
```

2. One of the downsides of QuickSort is that you have to be very careful writing the Partition method so that it handles duplicate pivots well. Here is one way to approach it. Have the Partition method divide the array into THREE pieces rather than two: Elements before the pivot, elements equal to the pivot, and elements after the pivot. For example, if the array were {I,H,A,T,E,T,A,C,O,S} and the random pivot turned out to be T, the array could be partitioned into {I,H,A,E,A,C,O,S,T,T} (the T's are together, and nothing comes after the pivot in this case).

Northern Michigan University (Marquette Co, MI)

CS422-01-25W: Algorithms (Andrew A. Poe) Name: _____
Midterm Examination (Exam 1) Friday 28 February 2025 9:00 A.M. EST

Write the method `int[] Partition (String[] A, int first, int last, int piv){ ... }` where A is the array of String, [first...last] is the portion of the array that needs be partitioned and rearranged and piv is the index of a pivot that has already been chosen between first and last. (You don't have to find the random pivot yourself.) The method returns an array of size 3, containing the lengths of the before, equal, and after portions of the rearranged array (in the above example, it would be {8,2,0}. There IS a way to do it without creating extra arrays but you are ALLOWED to create extra arrays if you want to.

```
int[] Partition (String[] A, int first, int last, int piv) {

 String[] bef = new String[last-first+1];
 String[] eq = new String[last-first+1];
 String[] aft = new String[last-first+1];
 int bct = 0, ect = 0, act = 0;
 for (int i=first; i <= last; i++) {
  int cmp = A[i].compareTo (A[piv]);
  if (cmp < 0) bef[bct++] = A[i];
  else if (cmp==0) eq[ect++] = A[i];
  else aft[act++] = A[i];
 }
 for (int i=0; i < bct; i++) A[first+i] = bef[i];
 for (int i=0; i < ect; i++) A[first+bct+i] = eq[i];
 for (int i=0; i < act; i++) A[first+bct+ect+i] = aft[i];
 int[] ret = new int[3];
 ret[0] = bct; ret[1] = ect; ret[2] = act;
 return ret;
}
```

3. I have an array that looks like: Q F C A N Z. The first three elements (Q F C) form a heap. Add the next three elements (A N Z) one at a time into the heap, showing your steps. What does the array look like after these elements have been added?

Q F C A : A is at position 3. Its parent is at (3-1)/2 or 1. Since F > A, A is in the correct position.

Q F C A N : N is at position 4. Its parent is at (4-1)/2 or 1. Since N > F, we will swap:

Q N C A F : N is at position 1. Its parent is at (1-1)/2 or 0. Since Q > N, N is in the correct spot.

Q N C A F Z: Z is at position 5. Its parent is at (5-1)/2 or 2. Since Z > C, we will swap.

Q N Z A F C: Z is at position 2. Its parent is at (2-1)/2 or 0. Since C > Q, we will swa.

Z N Q A F C: Z is now at the top of the heap and can't move further.

Northern Michigan University (Marquette Co, MI)

CS422-01-25W: Algorithms (Andrew A. Poe) Name: _____
Midterm Examination (Exam 1) Friday 28 February 2025 9:00 A.M. EST

Z N Q A F C

4. Imagine that you are hired to write code for a legacy platform using the BASIC language (I have actually had to do this!!!) Here's what you need to know about BASIC. Recursion and pointers do not exist, but arrays do. Methods exist, but they are a pain in the neck. They don't have their own variables (scope does not exist) and you can't pass parameters to them. Consider the six sorts we discussed in class: Insertion, Merge, Shell, Tree, Quick, Heap. What are the pros and cons of coding each of these sorts in BASIC? Explain your answer; don't just say "Good" "Bad" etc.

INSERTION: It doesn't matter what you're language you're using. Insertion is too slow.

MERGE: Merge is heavily recursive, making it difficult to program in BASIC.

SHELL: No recursion, no extra array, and short code. This is a fine choice for BASIC, although not as fast as HEAP or TREE.

TREE: No recursion, but you need a fair bit of extra space to pull it off. If you have the extra space, this is about as fast as you can get in BASIC. (This was my preferred choice when I was programming in BASIC.)

QUICK: Quick is heavily recursive, making it difficult to program in BASIC. (I've done it, though, and it's a pain in the neck!)

HEAP: No recursion, no extra array, but complex code. It's still faster than SHELL but not as fast as TREE. For large arrays where you're not sure you'll have the extra space required by TREE, HEAP isn't a bad choice, and this is why it was invented in the first place. On modern computers, space isn't really a concern, and so HEAP is largely obsolete.