Northern Michigan University (Marquette Co, MI)

CS422-01-25W: Algorithms (Andrew A. Poe) Name: ______________________
Practice Final Examination Sunday 27 April 2025 7:00 P.M. EDT

Time: 50 minutes

1. Given the following classes:

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

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

Write code for the LL method `int DelPrefix (String pref) { ... }` This method removes all nodes from the linked list whose strings begin with pref. If pref were "BL" for example, the method would remove "BLACK" "BLEND" "BLONDE" etc. The method returns the total number of nodes deleted. The linked list is not necessarily sorted. All the remaining nodes must be in the same relative order that they were in previously. You are not to create new nodes or modify the data fields of existing nodes. This must be accomplished by pointer manipulation only. Do not use loops. Use recursion only. You may use helper methods but you must write them if you do.

```
LL:
int DelPrefix (String pref) {

 int ct = 0;
 if (head !=null) ct = head.DelPrefix (this,null,pref);
 return ct;
}

Node:
int DelPrefix (LL L, Node prev, String pref) {

 int ct = 0;
 if (next != null) ct =next.DelPrefix (L,this,pref);
 if (pref.equals(s.substring(0,pref.length()))) {
  if (prev != null) prev.setnext (next);
  else L.setnext (next);
  ct++;
 }
 return ct;
}
```

2. If A is an array of String in which all strings contain only capital letters, write `void SortByFirst (String[] A) { ... }` which sorts the strings by the first character only. So, all the A's are together, all the B's are together, etc. The thing is that the method should run in O(*n*) time on the number of strings in the array. *n* lg *n* is too slow. You may assume that the empty string does not appear in the array.

```
void SortByFirst (String[] A) {

 int[] ct = new int[26];
 for (int i=0; i < A.length; i++)
  ct[((int)A[i].charAt(0))-65]++;
 for (int i=1; i < ct.length; i++)
  ct[i]+= ct[i-1];
 String[] B = new String[A.length];
 for (int i=A.length-1; i>=0; i--) {
  B[--ct[((int)A[i].charAt(0))-65] = A[i];
 for (int i=0; i < A.length; i++)
  A[i] = B[i];
}
```

3. Show step by step how Heap Sort would sort F I N A L.

F I N A L

First we have to make it a heap.

Each position's parent (except 0) is (n-1)/2 rounded down.
Each position's children are 2n+1 and 2n+2.

L's parent is I. Out of order. Swap. F L N A I
A's parent is L. Fine.
N's parent is F. Out of order. Swap. N L F A I.
F has no children. Fine.
L's parent is N. Fine.
It is now a heap. Each node is greater than both of its children.

Swap the first and last elements. I L F A N. N is no longer in the heap.
I is out of order. Swap with L. L I F A N. Fine.
Swap the first and last elements. A I F L N. L is no longer in the heap.
A is out of order. Swap with I. I A F L N. Fine.
Swap the first and last elements. F A I L N. I is no longer in the heap.
F is not out of order, so fine.
Swap the first and last elements. A F I L N. F is no longer in the heap.
There is now only one element in the heap. So we're done. A F I L N.

4. Write the code for the following method

```
public void FileShellSort (RandomAccessFile f, int reclen)
 { ... }
```

f is a random access file that is already open. You don't have to open or close it. reclen is the length of each individual record. You are to sort the file using the 2-3 Shell Sort described in class. For example, if the file were "CATDOGBAT" and the record length were 3, the method should transform the file into "BATCATDOG". You are not allowed to have more than two records in memory at any given time. You are not allowed to use an additional file.

```
public void FileShellSort (RandomAccessFile f, int reclen) {

 int rct = (int) (f.length()/reclen);
 for (int d = rct-1; d >= 1; d--) {
  int td=d;
  while (td%2==0) td/=2;
  while (td%3==0) td/=3;
  if (td==1)
   for (int i=d; i < rct; i++) {
    f.seek ((i-d)*reclen);
    byte ba = new byte[reclen];
    f.read (ba);
    String A = new String (ba);
    f.seek (i*reclen);
    f.read (ba);
    String B = new String (ba);
    if (A.compareTo(B) > 0) {
     f.seek ((i-d)*reclen);
     f.writeBytes (B);
     f.seek (i*reclen);
     f.writeBytes (A);
    }
   }
 }
}
```

5. Consider the following function f(0)=1, f(1)=2, f(2)=3, f(3)=4, f(4)=10, f(5)=20, f(6)=30, f(7)=40, f(8)=100, f(9)=200, f(10)=300, f(11)=400, etc. Using a hashtable and recursion, write the code for `BigInteger f (int n) { . . . }` which computes the value of this function.

Northern Michigan University (Marquette Co, MI)

CS422-01-25W: Algorithms (Andrew A. Poe) Name: ________________________________

Practice Final Examination Sunday 27 April 2025 7:00 P.M. EDT

Let the hashtable and recursion to the work. The mathematical formulas in your code should be simple.

```
HashMap <String,String> HT = new HashMap ();

public BigInteger (int n) {

 BigInteger bi = null;
 String ans = HF.get (""+n);
 if (ans==null) {
  if (n<=3) bi = new BigInteger (""+(n+1));
  else bi = new BigInteger("10").multiply (f(n-4));
  HF.put (""+n,bi.toString());
 } else bi = new BigInteger (ans);
 return bi;
}
```

6. Given a trie, describe, in English, a recursive algorithm that will print all strings in the trie that match the search string once the X's have been removed from all strings in the trie. The search string will never contain an X. There might be X's in the trie. For example, if the search string were "CAT", this string would match "CAT" "CAXT" CAXXT" "XCXAXTX" "XXXXXXXCATXXXXX" for example, and these strings would print, if they are in the trie.

If my search string is empty, and I'm at an end-of-string marker, print and continue.
If my search string is empty, and I'm not at an end of string marker, don't print but continue.

If I have a child node that matches the first letter of the search string, recurse on that node with the first letter removed from the search string.
If I have a child node of "X" then recurse on that node with the search string unchanged.
You may well have to do both recursions.