Computer Science Foundation Exam

May 6, 2005

Computer Science

Section 1A

No Calculators!

KEY

Name: ______________________________

SSN: ______________________________

Score: /50

In this section of the exam, there are four (4) problems. <u>You must do all of them.</u>

The weight of each problem in this section is indicated with the problem. Partial credit cannot be given unless all work is shown and is readable.

Be complete, yet concise, and above all <u>be neat</u>.

1. [**12 pts**] Transform the following infix expression into its equivalent postfix expression using a stack. Show the contents of the stack at the indicated points A, B and C in the infix expressions.

A B C

R + P * N – D * (E / F – K) – L + M

A	B	C
	(	
*	*	
+	–	–

Grading: 3 points for each stack filled correctly (Total 9 points)

Resulting Postfix Expression :

R	P	N	*	+	D	E	F	/	K	–	*	–	L	–	M	+

Grading: 3 points for correct <u>post fix</u> expression (If order of operators is wrong but operands are in correct order , give 1 point).

2. Indicate the time complexity for each of the following operations in terms of Big-O notation, assuming that efficient implementations are used. Give the *worst case* complexities in terms of **n**, assuming that each data structure holds **n** elements.

a) Searching for an element in a sorted array using Binary Search. **O(log n)**
Grading: 3 points if correct , zero otherwise.

b) Push operation in a linked list stack containing n elements. **O(1)**
Grading: 3 points if correct , zero otherwise.

c) Deleting the first element in a circularly linked list APLPHA, where ALPHA points to the last element of the list **O(1)**
Grading: 3 points if correct , 1 point if wrong

d) Inserting a node in a binary search tree. **O(n)**
Grading: 3 points if correct , 1 point if wrong

e) Searching for a specific element in a linked list whose elements are in ascending order. **O(n)**
Grading: 3 points if correct , zero otherwise.

f) Adding 25 to each node of a Balanced Binary Search Tree. **O(n)**
Grading: 3 points if correct , 1 point if wrong

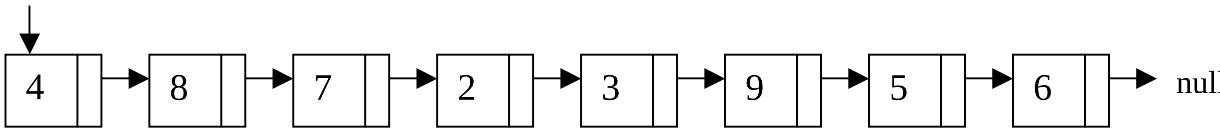
g) Checking if FOUR lists, each of size n are identical to each other. **O(n)**
Grading: 3 points for O(n)
1 point for O(4n)
Zero otherwise

h) Applying quicksort on an array whose elements are ALREADY SORTED in the proper order. **O(n^2)**

Grading: 3 points if correct , zero otherwise

3. [6 points] Consider the linked list shown below where pList points to the node containing the value 4.

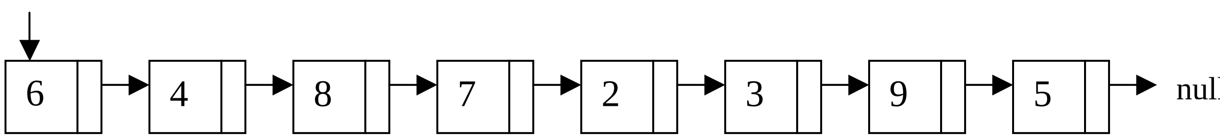
pList



Draw the list again showing the changes after the following code is executed.

```
pCur= pList;
     while ( pCur->next->next != NULL)
          pCur = pCur->next;
     pCur->next->next = pList;
     pList = pCur->next;
     pCur->next = NULL;
     pCur = NULL;
```

SOLUTION:
pList



Grading: 6 points if new list is as shown above.
Only 3 points if new list is same as above except that it does not contain node 5 at the end
Only 1 point if the list is anything else

[**8 pts**] Work out the number of operations involved in the following function msort. It is specified that n is a power of 2 and that `line12` needs n operations. (Note that n1 will always equal n2.) Ignore all operations involved in `line3` to `line9` from your calculations.

```
void msort(int array[], int n) {                          line 1

  int j,n1,n2,leftarray1[n],rightarray[n];                line2
  if (n<=1)return;                                        line3
       n1=n/2;                                            line4
  n2 = n - n1;                                            line5
  for ( j = 0; j<n1; j++ )                                line6
      leftarray[j]= array[ j];                            line7
  for ( j = 0; j<n2; j++ )                                line8
      rightarray[j]= array[ j+n1];                        line9
  msort(leftarray, n1);                                   line10
  msort(rightarray, n2);                                  line11
  merge(leftarray, n1, rightarray, n2,  array, n);line12
}
```

Recurrence relation for the function:

$T(1) = 1$
$T(n) = T(n/2) + T(n/2) + n$ (since n is a power of 2, $n1 = n2 = n/2$)
Grading: 3 points if recurrence relation is correct
Using same logic and going further down

$T(n/2) = 2\,T(n/4) + n/2$

Substituting for $T(n/2)$ in the equation for $T(n)$,we get

$T(n) = 2[\,2\,T(n/4) + n/2\,] + n$

$= 4\,T(n/4) + 2n$
Grading: 1 point more if solution is correct up to this point.

Again by rewriting $T(n/4)$ in terms of $T(n/8)$, we have

$T(n) = 4\,[\,2\,T(n/8) + n/4\,] + 2n$

$= 8\,T(n/8) + 3\,n$

$= 2^3\,T(n/2^3) + 3\,n$

The next substitution would lead us to

$T(n) = 2^4 \; T(n/ 2^4) + 4 n$

Continuing in this manner, we can write for any k,

$T(n) = 2^k \; T(n/ 2^k) + k n$

Grading: 2 points more if solution is correct up to this point.

This should be valid for any value of k. Now setting $2^k = n$. , gives k =log n.
Substituting in the expression for T(n)

$T(n) = n \, T(1) + n \log n$

$= n \log n + n$

Grading: 2 points more if final solution is correct.