Computer Science Foundation Exam

August 13, 2010

Section I B

COMPUTER SCIENCE

NO books, notes, or calculators may be used,
and you must work entirely on your own.

Name: KEY

PID: ______________________

Question #	Max Pts	Category	Passing	Score
1	10	ANL	7	
2	10	DSN	7	
3	10	DSN	7	
4	10	ALG	7	
5	10	ALG	7	
TOTAL	50			

You must do all 5 problems in this section of the exam.

Problems will be graded based on the completeness of the solution steps and not graded based on the answer alone. Credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all be neat.

1) (10 x 1 pt): Analysis

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. Following notations are being used:

AINC is an array containing **n** integers arranged in increasing order.
AD is an array containing **n** integers arranged in decreasing order.
AR is an array containing **n** integers in random order.
Q is a queue implemented as a linked list and containing **p** elements.
LINK is a linked list containing **n** nodes.
CIRC is a circular linked list containing **n** elements, where **C** points to the last element.
T is a binary search tree containing **n** nodes.

a) Searching for an element in AINC using linear search. ___ **O(n)** _____

b) Deleting the 10th node of linked list LINK. ____**O(1)** _____

c) Calling a function which uses Q, and calls *dequeue* **m** times. ___ **O(m)** ______

d) Inserting an element at the end of the list CIRC. ____ **O(1)** _____

e) Deleting the last element of CIRC. ____ **O(n)** _____

f) Finding the largest element of T. _____**O(n)** _____

g) Determining the height of T. ____ **O(n)** _____

h) Making the call selectionsort (AINC, n). ____ **O(n^2)** _____

i) Making two calls one after another. The first call is mergesort(AD,n), followed by the call insertionsort(AD,n). _____ **O(nlgn)** _____

j) Converting a decimal integer *num* into its binary equivalent. ___ **O(log num)** ______

Grading: 1 pt each no partial credit.

2) (10 points) Binary Trees

Write a recursive function that will find the height of a binary tree. The height of an empty tree is defined as -1. The height of a single node tree is defined as 0.

```
struct treeNode {
   int data;
   struct treeNode *left, *right:
};
```

```
int height (struct treeNode *ptr)  {
```

One possible solution is:

```
int height (struct treeNode *ptr)
{
     int leftheight, rightheight;

     if (ptr == NULL) // 1 pt
         return -1;   // 1 pt
     else
     {     leftheight = height(ptr->left); // 2 pts
           rightheight = height(ptr->right); // 2 pts
           if (leftheight > rightheight)  // 2 pts
               return(leftheight + 1); // 1 pt
           else
               return(rightheight + 1); // 1 pt
     }
}
```

```
}
```

3) (10 points) Linked Lists

Write a function which accepts a linear linked list J and converts it into a circular linked list. The function should return a pointer to the last element. The function prototype is provided for you below.

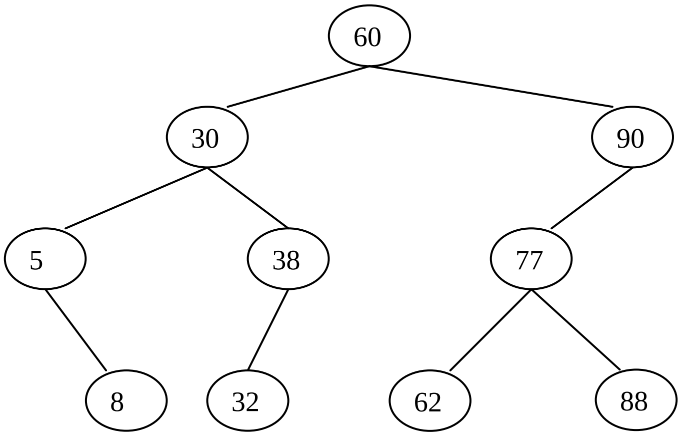
The node structure is as follows:

```
struct listNode {
     int data;
     struct listNode *next;
};
```

```
struct listNode * convert ( struct listNode * J)
{
    if (J == NULL)  // 1 pt
        return NULL; // 1 pt
    struct listNode * temp = J; // 1 pt
    while ( temp -> next != NULL) // 2 pts
        temp = temp->next;      // 2 pt
    temp->next = J;  // 2 pts
    return temp;  // 1 pt
}
```

4) (10 points) Binary Trees

Given the binary tree shown below, determine the order in which the nodes of the binary tree shown above are visited assuming the function **A(root)** is invoked. Assume that the tree nodes and pointers are defined as shown. Assume that **root** is a pointer to the node containing 60. Place your answers in the boxes provided.



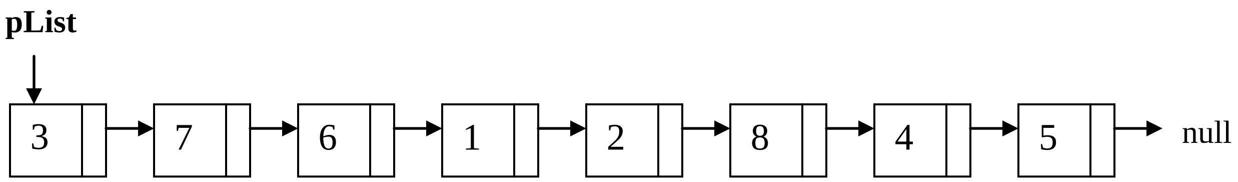
```
struct treeNode{
   int data;
   struct treeNode *left, *right:
}
struct treeNode *tree_ptr;

void A(struct treeNode *node_ptr){
    if (node_ptr != NULL){
       printf(“%d ,”,node_ptr->data);
       B(node_ptr->left);
       B(node_ptr->right);
    }
}

void B(struct treeNode *node_ptr){
    if (node_ptr != NULL) {
       A(node_ptr->left);
       printf(“%d ,”,node_ptr->data);
       A(node_ptr->right);
   }
}
```

ANSWER:

60	**5**	**8**	**30**	**38**	**32**	**77**	**62**	**88**	**90**

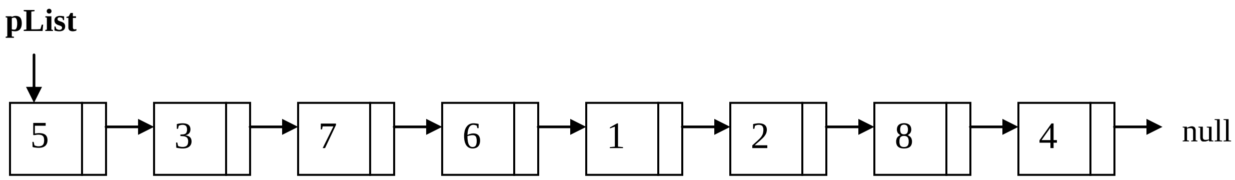
Grading: 1 pt per slot, no partial credit.

5) (10 points) Linked Lists Consider the linked list shown below where `pList` points to the node containing the value 3. Redraw the list showing the changes to the list 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:



Grading: pList pointing to 5 (2 pts)
5 attached to 3 (3 pts)
4's next being null (3 pts)
Whole list being intact (2 pt)