Instructions
1. Do all of the 4 problems
3. You have 50 minutes for the exam
4. Show all your work
5. Do not separate midterm papers
1. (25 pts) List the sequence of nodes visited by traversals of the following tree

(a) List the sequence of nodes visited by *preorder* traversal.

**Solution:** ABDGEHICFJ

(b) List the sequence of nodes visited by *inorder* traversal.

**Solution:** GDBHEIACJF

(c) List the sequence of nodes visited by *postorder* traversal.

**Solution:** GDHIEBJFCA
2. (25 pts) Write a **RECURSIVE** delete function for linked lists that deletes the first occurrence of a given integer from the list and returns the resulting list. Node structure is given below.

```c
struct node
{
    int info;
    struct node *next;
};
typedef struct node node;
```

Solution:

```c
node *delete(node *ptr, int i)
{
    if (ptr == NULL)
        return (ptr);

    if (ptr->info == i)
    {
        node *ptr2 = ptr->next;
        free(ptr);
        return (ptr2);
    }

    ptr->next = delete(ptr->next,i);

    return(ptr);
}
```
3. (25 pts) Write an **ITERATIVE** function to find the lowest common ancestor in a binary search tree and return a pointer. Lowest common ancestor between two nodes $a$ and $b$ in tree $T$ is defined as the lowest node in $T$ that has both $a$ and $b$ as descendants. Node structure is given below.

```c
struct node {
    int key;
    struct node *left, *right;
};
typedef struct node node;
```

Consider the following tree as an example. lowest common ancestor of 4 and 8 is 6. lowest common ancestor of 4 and 14 is 10.

![Tree Diagram]

**Solutions:**

```c
node *lca(node *ptr, int a, int b)
{
    while (ptr != NULL)
    {
        if (ptr->key >= a && ptr->key <= b)
            return (ptr);
        else if (ptr->key > b)
            ptr = ptr->left;
        else if (ptr->key < a)
            ptr = ptr->right;
    }
    return (ptr);
}
```
4. (25 pts) Minimum depth of a binary search tree is the shortest path from root node down to the nearest leaf node. Write a **RECURSIVE** function `mindepth()` to compute the minimum depth a binary search tree. You can use `min()` function that returns the minimum of two parameters in your implementation. Node declaration of the tree is given below.

```c
struct node {
    int key;
    struct node *left, *right;
};
typedef struct node node;
```

Minimum depth of the following tree is 2. Shortest path is from node 10 to 14.

```
4 8  
6 14
10   
```

Solution:

```c
int mindepth(node *ptr)
{
    if (ptr == NULL)
        return 0;
    else
        return (1+min(mindepth(ptr->left),mindepth(ptr->right)));
}
```