Name:          KEY & GRADING POLICY

SSN: ____________________________

<table>
<thead>
<tr>
<th>Max Pts</th>
<th>Type</th>
<th>Passing Threshold</th>
<th>Student Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>10</td>
<td>ANL</td>
<td>7</td>
</tr>
<tr>
<td>Q2</td>
<td>10</td>
<td>ANL</td>
<td>6</td>
</tr>
<tr>
<td>Q3</td>
<td>10</td>
<td>DSN</td>
<td>7</td>
</tr>
<tr>
<td>Q4</td>
<td>10</td>
<td>DSN</td>
<td>7</td>
</tr>
<tr>
<td>Q5</td>
<td>10</td>
<td>ANL</td>
<td>6</td>
</tr>
<tr>
<td>Total</td>
<td>50</td>
<td></td>
<td>33</td>
</tr>
</tbody>
</table>

You must do all 5 problems in this section of the exam. Partial credit cannot be given unless all work is shown and is readable.

Be complete, yet concise, and above all be neat. Do your rough work on the last page.
1. [10 pts]
Indicate the worst case time complexity in terms of Big-O for each of the following operations:

a) Searching for a target in a sorted array containing N integers
   ____O(log N)_____

b) Calling a function which returns the data stored in the last node of a linked list containing M nodes.
   ____O(M)_____

c) Searching for a target in a Binary Search Tree containing N nodes.
   ____O(N)_____

d) Adding 5 to all leaf nodes of a binary tree containing P integers
   ____O(P)_____

e) Calling the POP function M times on an array based stack containing N items
   ____O(M)_____

f) Inserting a node in a sorted linked list containing P nodes
   ____O(P)_____

g) calling a function to return the largest element of a sorted array containing N integers
   ____O(1)_____

h) running the following sorting algorithms on an array containing N integers already sorted in the correct order

1. Selection sort
   ____O(N^2 )_____

2. Merge sort
   ____O(N log N)____

3. Quick sort (first element as pivot)
   ____O(N^2 )_____

2.a) [5 pts]
Use summations to express the final value of beta being returned by the following function. You are not required to solve the summations.

```c
int compute ( int n)
{
    int beta = 0;
    int alpha = 14;
    for ( k= 1 ;    k <= 20  ;   k++)
        for ( j = n;   j < 2n+1 ;  j++)
            beta = beta + alpha *j – 6*n ;
    return beta;
}
```

\[ \sum_{k=1}^{20} \sum_{j=n}^{2n} 14j – 6n \]

b) [5 pts] Write the recurrence relation to indicate the total number of operations T(n), needed to execute the following function in terms of n. Consider the worst case. You are not required to solve the recurrence relation.

```c
int  newfunction(int AA[ ], int n)
{
    if (n == 1) return 0;
    if (AA[n-1] > 50)
        return 1 + newfunction(AA,n-1);
    else
        return newfunction(AA, n-1);
}
```

\[ T(n) = T(n-1) + 2 \]  
or  
\[ T(n) = T(n-1) + 1 \]
3. [10 pts] Write a function which returns a linked list α after deleting its last node. The node structure is given below.

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

struct node * deletelast ( struct node * alpha)
{
    struct node * cur, *pre;
    // proper setting of two pointers [2 pts]
    pre = alpha;
    cur = alpha->next;

    // correct WHILE loop [2 pts]
    while ( cur->next != NULL){
        cur = cur ->next;
        pre = pre->next;
    }

    // terminate list at right place [2 pts]
    pre->next = NULL;

    // free the deleted node [1 pt]
    free(cur);

    // return the list [1 pt]
    return alpha;
}
struct treenode * insert74 ( struct treenode * p) 
{
    if( p== null)
    {
        p = (struct treenode*)(malloc(sizeof(struct treenode)));
        p->data = 74;
        p->left = NULL;
        p->right = NULL;
    }
    else if( p->data > 74)
        p->left = insert74( p->left);

    else if( p->data < 74)
        p->right = insert( p->right );

    return p;
}
5. a) [5 pts] Indicate in few words, the purpose of the following function.

```c
int newfunction (struct treenode * p){
    if(p!=NULL){
        if( (p->left!= NULL   !=     p->right!=NULL)){
            if(p->data %2 == 1)
                return 1 + newfunction (p->left) +
                     newfunction (p->right);
        }
        else
            return newfunction (p->left) +
                   newfunction (p->right);
    }
}
```

The function counts the number of single-child nodes with odd values.

b) [5 pts] What does the function return, given the following tree?

![Tree Diagram]

It returns 4 (corresponding to nodes 11, 51, 17, 87)