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

Partial credit cannot be given unless all work is shown and is readable.

Be complete, yet concise, and above all **be neat.**
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 1, 2 and 3 in the infix expressions.

\[
\begin{align*}
1 & \quad 2 & \quad 3 \\
P & / (B + S - T) + M * N / C & - A + F \\
\end{align*}
\]

Resulting Postfix Expression:

\[
P B S + T - / M N * C / + A - F +
\]
2. [11 x 2 pts] 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.

2 points each correct answer
0 otherwise

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) Doubling the value stored in root node of T. ___ O(1) ___
h) Making the call selectionsort (AINC, n). ___ O(n^2) ___
i) Making the call bubblesort (AINC, n). ___ O(n) ___
j) Making two calls one after another. The first call is mergesort(AD,n), followed by the call quicksort(AD,n). ___ O(n^2) ___
k) Converting a decimal integer num into its binary equivalent. ___ O(log num) ___
3. [8 pts] 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 node structure is as follows:

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

Partial credits may be awarded if the function is not correct.

```c
struct node * convert ( struct node * J) {
    struct node * temp = J;
    while ( temp -> next != NULL) {
        temp = temp->next;
        temp->next = J;
    }
    return temp;
}
```
4. [8 pts] Write the recurrence relation for the following function which takes as input the first \( n \) elements of the array \( \text{ARRAYS} \) holding integers. Solve it to work out the total number of operations, and the time complexity of the algorithm.

```c
int modify( int \text{ARRAYS}[ ], int n) {
    int maximum;
    if (n==1) return \text{ARRAYS}[0];
    else {
        maximum = findmax(\text{ARRAYS}, n);
        \text{ARRAYS}[n-1] = maximum;
        return modify(\text{ARRAYS}, n-1);
    }
}
```

The function \( \text{findmax} \) returns the largest value in the array \( \text{ARRAYS} \) of size \( n \).

```c
int findmax(int A[ ], n) {
    max = A[0];
    for (i=1, i<=n, i++)
        if (A[i] > max) max=A[i];
    return max;
}
```

The \( \text{findmax} \) function has a time complexity of \( O(n) \).

So it can simply be taken as \( n \)

Recurrence relation:

\[ T(n) = T(n-1) + n \]

\[ T(1) = 1 \]

\[ T(n-1) = T(n-2) + n-1 \]

Substituting back \( T(n) = T(n-2) + n + n-1 \)

Again \( T(n-2) = T(n-3) + n-2 \)

Substituting back \( T(n) = T(n-3) + n + n-1 + n-2 \)

In general

\[ T(n) = T(n-k) + n + n-1 + \ldots + (n-k+1) \]

Since \( T(1)=1 \), Let \( n-k = 1 \)
This gives \( k = n-1 \)
Thus \( T(n) = T(1) + n + n-1 + \ldots + 3 + 2 \)

\[ = n + n-1 + \ldots + 3 + 2 + 1 \]

\[ = n(n +1)/2 \]
Time complexity = \( O(n^2) \)