DAA Lab Programs Analysis

This document provides detailed explanations, corrected code, outputs, challenging code parts, and pseudocode for each of the 10 DAA lab programs (including sub-programs where applicable).

Program 1: Binary Search using Recursion

1. Explanation:

Binary Search is a divide-and-conquer algorithm used to find the position of a target value within a sorted array. It compares the target value to the middle element and recurses on the half in which the target might exist.

2. Corrected Code:

```
#include <stdio.h>
int BinarySearch(int array[], int start_index, int end_index, int element) {
   if (end_index >= start_index) {
      int middle = start_index + (end_index - start_index) / 2;
      if (array[middle] == element)
         return middle;
      if (array[middle] > element)
         return BinarySearch(array, start_index, middle - 1, element);
      return BinarySearch(array, middle + 1, end_index, element);
   }
```

```
    return -1;
}

int main() {
    int array[20], n, element, i;
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    printf("Enter elements in sorted order: ");
    for(i = 0; i < n; i++) {
        scanf("%d", &array[i]);
    }
    printf("Enter key value: ");
    scanf("%d", &element);
    int found = BinarySearch(array, 0, n - 1, element);
    if(found == -1)
        printf("Element not found in the array\n");
    else
        printf("Element found at index: %d\n", found + 1);
    return 0;
}
```

3. Output:

```
Enter the number of elements: 5
Enter elements in sorted order: 1 2 3 4 5
Enter key value: 3
Element found at index: 3
```

4. Difficult Parts Explained:

- Correct middle calculation to avoid overflow.

- Recursion must reduce problem size correctly.

5. Pseudocode:

```
FUNCTION BinarySearch(array, start, end, key):
    IF start > end:
        RETURN -1
    mid = start + (end - start) / 2
    IF array[mid] == key:
        RETURN mid
    ELSE IF array[mid] > key:
        RETURN BinarySearch(array, start, mid - 1, key)
    ELSE:
        RETURN BinarySearch(array, mid + 1, end, key)
```