**Binary Trees**

**3A. Binary Tree:** Implement a program to construct a Binary Search Tree and perform insert, delete, and search operations. [This is implemented using a linked list]

**Algorithm:**

Step 1: Create class Student to store the details of the students with attributes reg no, first name, last name, phone number, and CGPA.

Step 2: Input the number of students ‘n’.

Step 3: Initialize an array of type Student of length n.

Step 4: Input the details of n students and store it in the array initialized above.

Step 5: Sort the given array using merge sort.

Step 6: For implementing merge sort, recursively divide the array into two sub arrays, and when the base case is reached, merge the sorted subarrays, leading up the sorted main array.

Step 7: Input the search query.

Step 8: Binary search the sorted main array for the query.

Step 9: Binary search is implemented by the following method:



**Program:** [In next page]



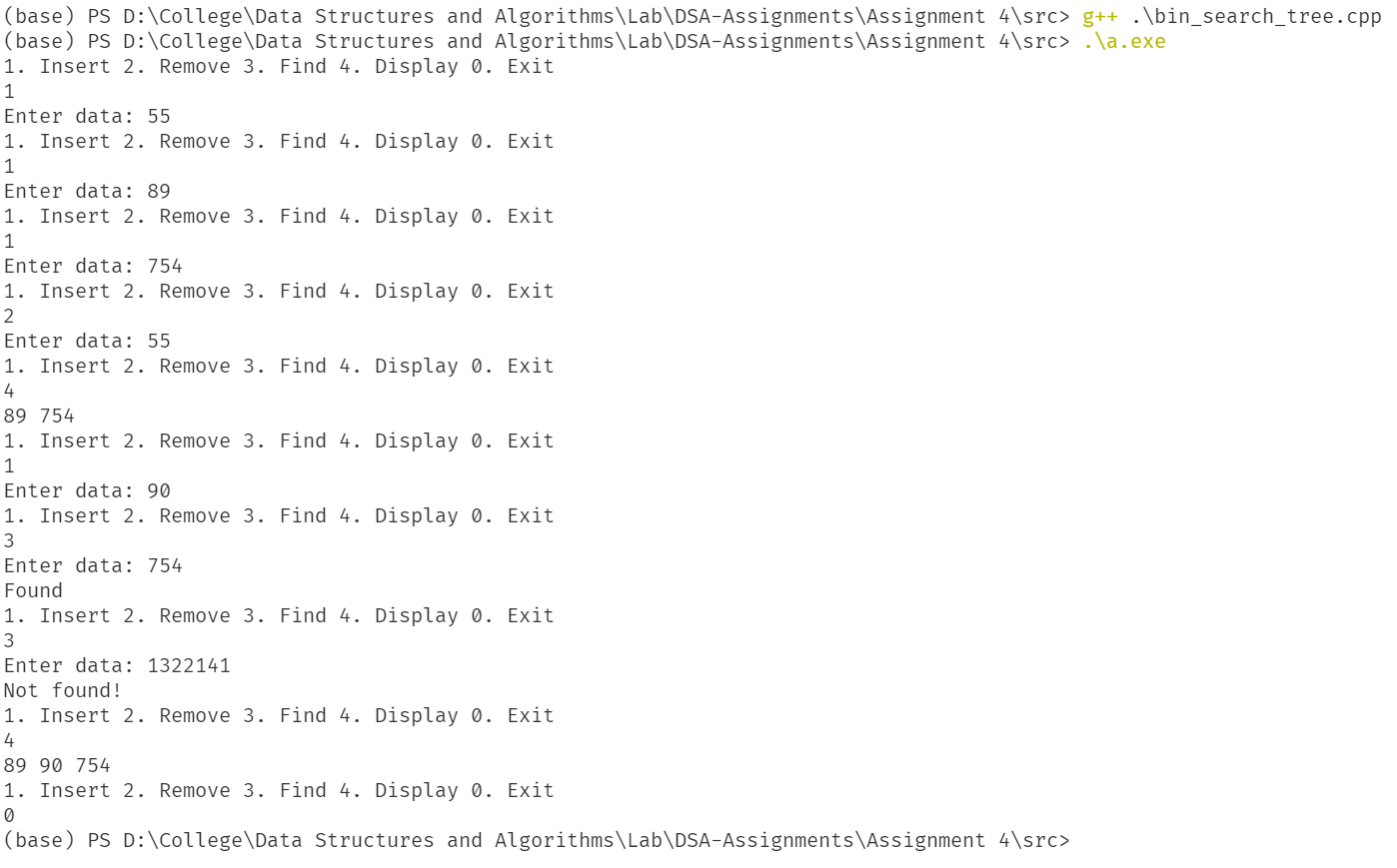








**Output:**

****

**Results:**

Thus, the binary search tree with insert, delete, find and display operations has been implemented.

**4B. Min and Max in BST:** Find the minimum and maximum elements in a binary search Tree

**Algorithm:**

Step 1: Input the number from the user.

Step 2: Call the square root function which is implemented in the following way:



Step 3: Print the value returned by the function.

**Program:** [In next page]

****

****

****

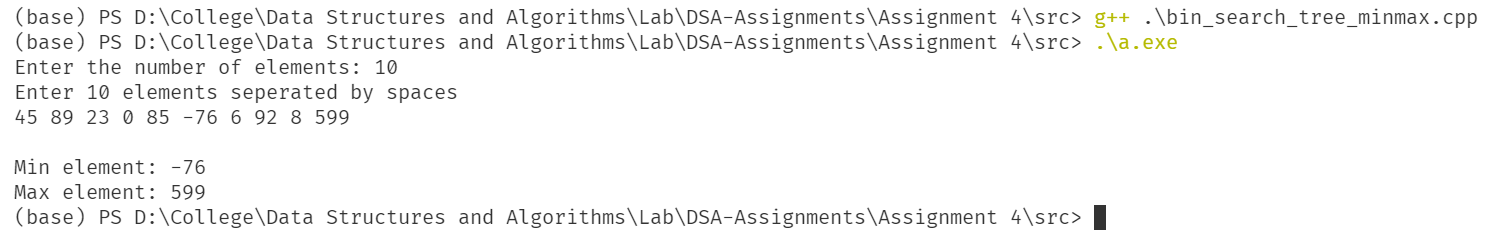
****

****

[Continued in next page]

****

**Output:**



**Results:**

Thus, the program to find minimum and maximum elements in a binary tree is implemented.

**[Note: Since question c and d were very similar, I have merged them into a single program]**

**4C | D. Expression Tree:** Implement a program to create an expression tree given an infix expression and get the prefix **and** postfix form of the expression using **pre-order** and **post-order** traversal respectively.

**Algorithm:**

Step 1: Input the number of elements in an array.

Step 2: Initialize an array of appropriate data type.

Step 3: Input the elements of the array

Step 4: Call bubble sort on the array which is implement as follows:



Step 5: Print the sorted array, the number of comparisons and the number of array accesses.

**Program:** [In next page]



[Continued in next page]

****

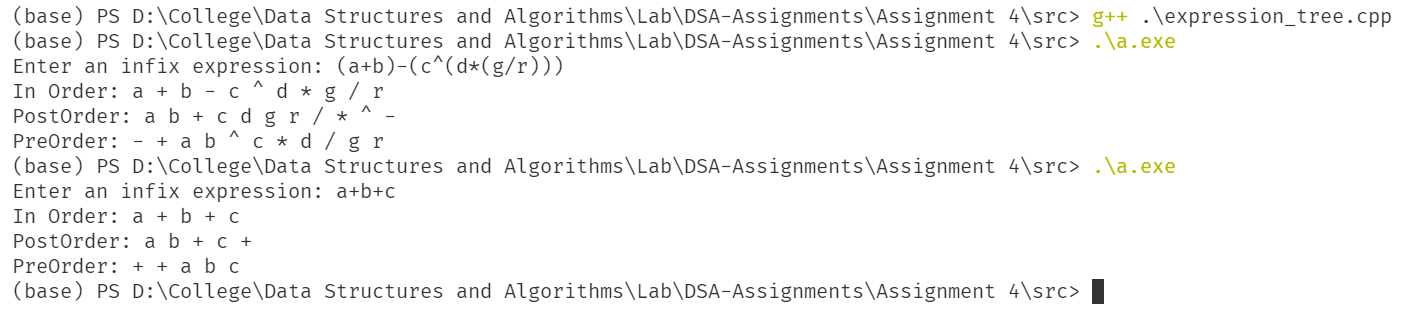
****

****

****

****

**Output:**

****

**Results:**

Thus, the program to build and traverse an expression tree from a given infix expression is implemented.