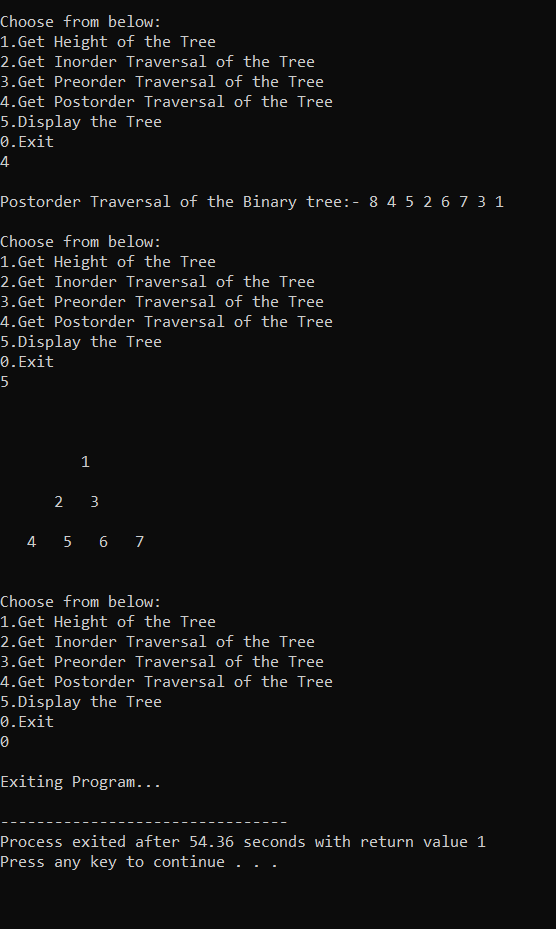
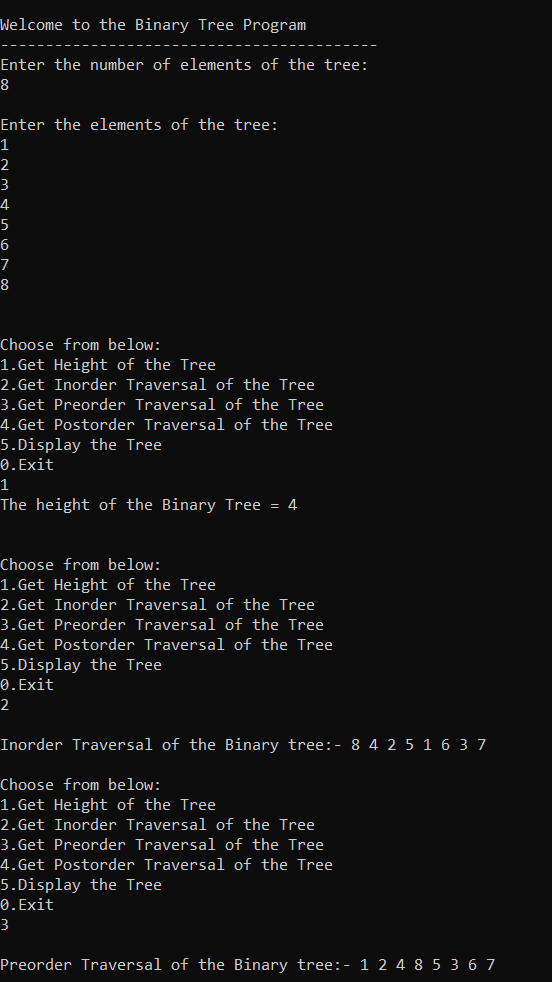
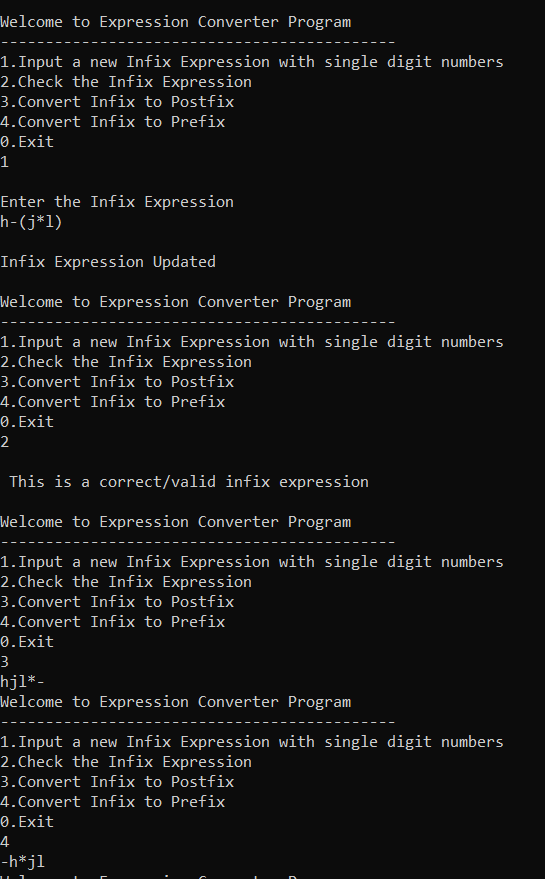
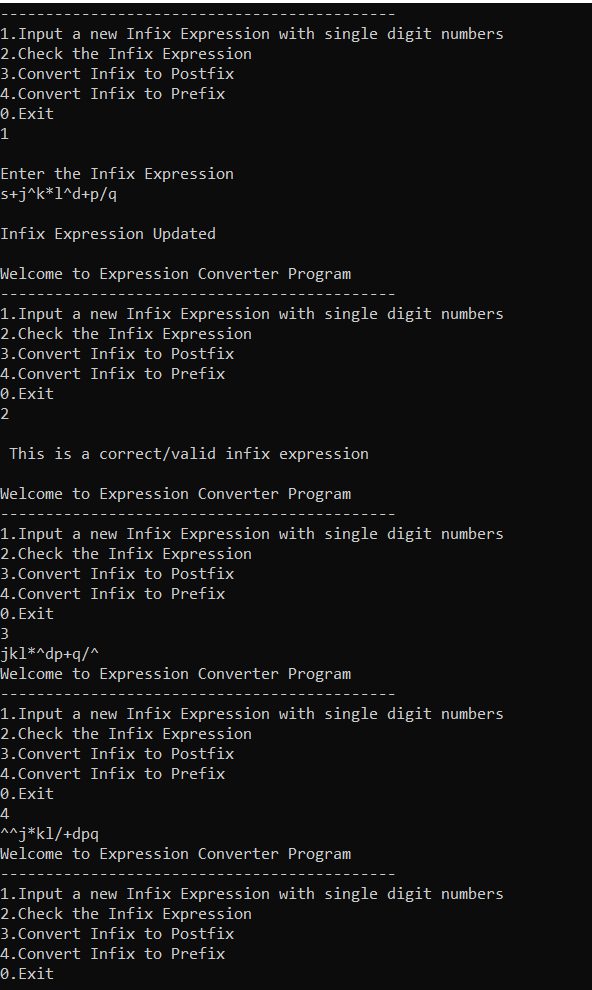
**Question 1: Create a Binary Tree, Calculate Height, Print Preorder,Inorder and Postorder Traversal, Print in Tree form.**

* The problem is solved by making use of arrays
* Program is written on C with three basic functions.
* (i) **checkTree(int x):** is used to check for my binary tree to have only two children the left and right child.
* (ii) **createTree(struct node\* root,int data)**: creates my Binary Tree from the input array of elements
* (iii) **heightTree(struct node \*root)**: calculates the height of my Binary Tree.
* (iv) **inOrder(struct node \*root):** traverses my Binary Tree in Inorder traversal and prints all the nodes.
* (v) **preOrder(struct node \*root):** traverses my Binary Tree in Preorder traversal and prints all the nodes.
* (vi) **postOrder(struct node \*root):** traverses my Binary Tree in Postorder traversal and prints all the nodes.
* (vii) **treeStage(struct node \*root, int i):** used by my printTree() while printing the tree in tree form.
* (vii) **printTree(struct node \*root):** Takes in the root of th tree as parameter and prints the Binary Tree in tree form.
* (viii) **main()**: main driver function that helps the user make use of all the functionalities provided by the program.
* The input is checked against any exception, by using if and else to check for expected input.  
  If unexpected input is received, The program returns Invalid Input and exits.

**Execution Screenshots:  
**

**Question 2: Take Infix Expression from the user, validate the infix expression, convert to prefix, convert to postfix expression**

* The problem is solved by making use of arrays
* Program is written on C with three basic functions.
* (i) **characterPriority(char ch):** is used to check the priority of my operator (operator precedence) for the expression and is used by the createTree() and checkInfix().
* (ii) **isOperator(char ch)**: checks if a certain character is a valid operator or not, a valid operator in this program can be either of these: + - \* / ^ ( )
* (iii) **createTree(char\* str)**: takes in the input string and traverses through every character and creates a binary expression tree for the given infix expression, makes us of two stack arrays, one of character and one of node type.
* (iv) **checkInfix(struct node \*root):** traverses my binary tree created by createTree() and checks if the given expression was valid or not, the expression will be valid if all the leaf nodes are alphanumeric or alphabets only and all the non leaf nodes can only be operators. If this is satisfied, my Infix expression is valid else it is invalid
* (v) **infixPostfix(struct node \*root)**: traverses my binary tree in postorder traversal and prints the resulting postfix expression
* (vi) **infixPrefix(struct node \*root)**: traverses my binary tree in preorder traversal and prints the resulting prefix expression
* (vii) **main()**:main driver function that helps the user make use of all the functionalities provided by the program.
* The input is checked against any exception, by using if and else to check for expected input.  
  If unexpected input is received, The program returns Invalid Input and exits.

**Execution Screenshots:  
** ****