NATIONAL INSTITUTE OF TECHNOLOGY PUDUCHERRY

(An Institution of National Importance under MHRD, Govt. of India)

KARAIKAL – 609 609

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Roll Number: CS19B1009 **Name** ARUN KUMAR R

Semester: II **Class:** B TECH - CSE

Subject Code: CS106 **Subject Name:** DATA STRUCTURES LABORATORY

TRAVERSAL OF TREE

DATE:10/03/20

AIM:

To traverse the given tree using inorder, preorder, postorder traversal.

ALGORITHM:

1. Start the program.
2. Get the tree from the user.
3. Declare the recursive functions for inorder(left child, root, right child), postorder(left child, right child, root), preorder(root, left child, right child).
4. Get the input from the user.
5. Output the result.
6. End the program.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
```

```
    int value;
    struct node *left, *right;
    int sub_height ;
};

NODE *create_node(int value)
{
    NODE *temp = (NODE *)malloc(sizeof(NODE));
    temp->left = temp->right = NULL;
    temp->value = value;
    temp->sub_height = 0;
    return temp;
}
int getInput(int *value, int arr[], int flag)
{
    if (flag == 0)
        printf("Enter the value\n");
    if (flag == 1)
        printf("\nEnter the operation to perform(help - 0)\n");
    if (flag == 2)
    {
        int n;
        printf("Enter the no of elements:\n");
        scanf("%d", &n);
        printf("Enter the elements:\n");
        for (int i = 0; i < n; i++)
            scanf("%d", &arr[i]);
        arr[n] = '\0';
        return 1;
    }
    if (flag == 3)
        printf("your option :\n");
    scanf("%d", value);
    return 1;
}
int insert_bst(NODE **root, int value)
{
```

```
    if (*root == NULL)
    {
        *root = create_node(value);
        return 1;
    }
    if ((*root)->value > value)
    {
        if ((*root)->left == NULL)
            (*root)->left = create_node(value);
        else
            insert_bst(&(*root)->left, value);
    }
    else if ((*root)->value < value)
    {
        if ((*root)->right == NULL)
            (*root)->right = create_node(value);
        else
            insert_bst(&(*root)->right, value);
    }
    return 1;
}

int *print(NODE **root)
{
    printf("%d\t", (*root)->value);
}
int in_order(NODE **root, int *(*callback)(NODE **))
{
    if (*root == NULL)
        return 0;
    in_order(&(*root)->left, callback);
    callback(root);
    in_order(&(*root)->right, callback);
    return 1;
}
int pre_order(NODE **root, int *(*callback)(NODE **))
```

```
{
   if (*root == NULL)
      return 0;
   //Node_count = callback(root);
   callback(root);
   pre_order(&(*root)->left, callback);
   pre_order(&(*root)->right, callback);
   return 1;
}
int post_order(NODE **root, int *(*callback)(NODE **))
{
   if (*root == NULL)
      return 0;
   post_order(&(*root)->left, callback);
   post_order(&(*root)->right, callback);
   callback(root);
   return 1;
}
int traverse(NODE **root)
{
   if (*root == NULL)
   {
      printf("ROOT IS NULL\n");
      return 0;
   }
   int i;
   printf("Enter the traversal method: inorder :1 preorder : 2 postorder: 3\n");
   while (1)
   {
      scanf("%d", &i);
      printf("\n");

      switch (i)
      {
      case 1:
         in_order(root, print);
         return 1;
```

```
        case 2:
            pre_order(root, print);
            return 1;
        case 3:
            post_order(root, print);
            return 1;
        default:
            printf("incorrect input, please try again.\n");
            break;
        }
    }
}
```

```
int binary_search_tree(NODE **root) //BST DRIVER
{
    printf("\n\nBINARY SEARCH TREE\n\n");
    int input, value;
    NODE *result;
    help(1);
    while (1)
    {
        printf(“help – 0, insert – 1, traverse – 2, exit - 3”);
        getInput(&input, NULL, 1);
        switch (input)
        {
        case 0:
            printf(“help – 0, insert – 1, traverse – 2, exit - 3”);
            break;
        case 1:
            getInput(&value, NULL, 0);
            if (insert_bst(root, value))
                printf("NODE WAS INSERTED\n");
            else
```

```
            printf("THE NODE WAS NOT INSERTED\n");
        break;
    case 2:
        traverse(root);
        break;
    case 3:
        printf("\nExited from BST\n\n");
        help(0);
        return 1;
```

OUTPUT:

```
insert : 1 delete : 2 search : 3 traverse : 4 exit : 5 max : 6 min : 7

Enter the operation to perform(help - 0)
4 1
Enter the traversal method: inorder :1 preorder : 2 postorder: 3

7       9       34      56
Enter the operation to perform(help - 0)
4 2
Enter the traversal method: inorder :1 preorder : 2 postorder: 3

56      34      7       9
Enter the operation to perform(help - 0)
4 3
Enter the traversal method: inorder :1 preorder : 2 postorder: 3

9       7       34      56
```

RESULT:

The program was executed successfully.