LAB-8 (BINARY TREE)

CODE:

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

typedef struct node {
    int data;
    struct node *left;
    struct node *right;
} node;

node *create(int value) {
    node *newnode;
    newnode = (node *)malloc(sizeof(node *));
    newnode->left = NULL;
    newnode->right = NULL;
    newnode->data = value;
    return newnode;
}

node *insert(node *root, int value) {
    if (root == NULL)
        return create(value);

    if (value < root->data)
        root->left = insert(root->left, value);
    else if (value > root->data)
        root->right = insert(root->right, value);

    return root;
}

node *find_min(node *ptr) {
    node *current = ptr;

    while (current && current->left != NULL)
        current = current->left;

    return current;
}

node *delete(node *root, int key) {
    if (root == NULL)
        return root;

```

```
44      if (key < root->data)
45          root->left = delete(root->left, key);
46      else if (key > root->data)
47          root->right = delete(root->right, key);
48      else {
49          if (root->left == NULL) {
50              node *temp = root->right;
51              free(root);
52              return temp;
53          } else if (root->right == NULL) {
54              node *temp = root->left;
55              free(root);
56              return temp;
57          }
58
59          node *temp = find_min(root->right);
60          root->data = temp->data;
61          root->right = delete(root->right, temp->data);
62      }
63
64      return root;
65  }
66
67  void inorder_trav(node *root) {
68      if (root != NULL) {
69          inorder_trav(root->left);
70          printf("%d -> ", root->data);
71          inorder_trav(root->right);
72      }
73  }
74
75  void preorder_trav(node *root) {
76      if (root != NULL) {
77          printf("%d -> ", root->data);
78          preorder_trav(root->left);
79          preorder_trav(root->right);
80      }
81  }
```

```c
void postorder_trav(node *root) {
    if (root != NULL) {
        postorder_trav(root->left);
        postorder_trav(root->right);
        printf("%d -> ", root->data);
    }
}

int main() {
    node *root = NULL;
    int choice, value;

    while (1) {
        printf("\n--- Binary Search Tree Menu ---\n");
        printf("1. Insert a node\n");
        printf("2. Delete a node\n");
        printf("3. Inorder Traversal\n");
        printf("4. Preorder Traversal\n");
        printf("5. Postorder Traversal\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");

        if (scanf("%d", &choice) != 1) {
            printf("Invalid input. Exiting.\n");
            break;
        }

        switch (choice) {
            case 1:
                printf("Enter value to insert: ");
                scanf("%d", &value);
                root = insert(root, value);
                break;

            case 2:
                printf("Enter value to delete: ");
                scanf("%d", &value);
                root = delete(root, value);
                break;

            case 3:
                printf("Inorder Traversal: ");
```

```
125                 inorder_trav(root);
126                 printf("NULL\n");
127                 break;
128
129             case 4:
130                 printf("Preorder Traversal: ");
131                 preorder_trav(root);
132                 printf("NULL\n");
133                 break;
134
135             case 5:
136                 printf("Postorder Traversal: ");
137                 postorder_trav(root);
138                 printf("NULL\n");
139                 break;
140
141             case 6:
142                 printf("Exiting...\n");
143                 return 0;
144
145             default:
146                 printf("Invalid choice. Please try again.\n");
147         }
148     }
149
150     return 0;
151 }
```

```
--- Binary Search Tree Menu ---
1. Insert a node
2. Delete a node
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 1
Enter value to insert: 5

--- Binary Search Tree Menu ---
1. Insert a node
2. Delete a node
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 1
Enter value to insert: 2

--- Binary Search Tree Menu ---
1. Insert a node
2. Delete a node
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 1
Enter value to insert: 14

--- Binary Search Tree Menu ---
1. Insert a node
2. Delete a node
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 1
Enter value to insert: 6

--- Binary Search Tree Menu ---
1. Insert a node
2. Delete a node
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 1
Enter value to insert: 1000

--- Binary Search Tree Menu ---
1. Insert a node
2. Delete a node
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 2
Enter value to delete: 1000

--- Binary Search Tree Menu ---
1. Insert a node
2. Delete a node
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 3
Inorder Traversal: 2 -> 5 -> 6 -> 14 -> NULL

--- Binary Search Tree Menu ---
1. Insert a node
2. Delete a node
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 4
Preorder Traversal: 5 -> 2 -> 14 -> 6 -> NULL

--- Binary Search Tree Menu ---
1. Insert a node
2. Delete a node
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 5
Postorder Traversal: 2 -> 6 -> 14 -> 5 -> NULL

--- Binary Search Tree Menu ---
1. Insert a node
2. Delete a node
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 6
Exiting...

Process returned 0 (0x0)   execution time : 37.702 s
Press any key to continue.
```