Bansilal Ramnath Agarwal Charitable Trust’s

Vishwakarma Institute of Technology, Pune-37

(Anautonomous Institute of Savitribai Phule Pune University)

Department of Multidisciplinary Engineering

Division	**CS-A**
Batch	**B1**
Roll no.	**90**
Name	**Aditya Shrinivas Kurapati**

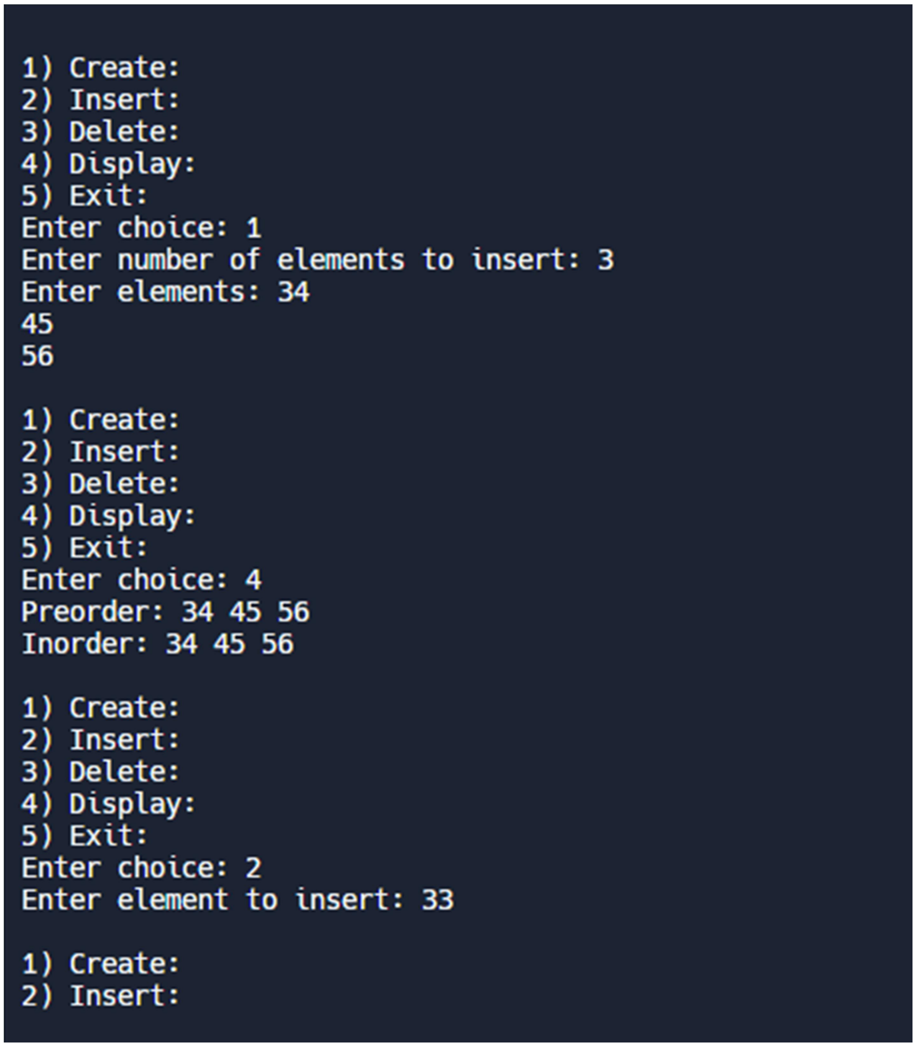
1. **Implement AVL TREE**

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

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

node *insert(node *T, int x);
node *Delete(node *T, int x);
int height(node *T);
node *rotateleft(node *x);
node *rotateright(node *x);
node *RR(node *T);
node *LL(node *T);
node *LR(node *T);
node *RL(node *T);
void preorder(node *T);
void inorder(node *T);

node *insert(node *T, int x) {
  if (T == NULL) {
    T = (node *)malloc(sizeof(node));
    T->data = x;
    T->left = NULL;
    T->right = NULL;
    T->ht = 1;
  } else if (x > T->data) {
    T->right = insert(T->right, x);
    if (height(T->right) - height(T->left) == 2) {
      if (x > T->right->data)
        T = RR(T);
      else
        T = RL(T);
    }
  } else if (x < T->data) {
    T->left = insert(T->left, x);
    if (height(T->left) - height(T->right) == 2) {
      if (x < T->left->data)
        T = LL(T);
      else
        T = LR(T);
    }
  }
  T->ht = height(T);
  return T;
}

node *Delete(node *T, int x) {
  node *p;
  if (T == NULL)
    return NULL;
  if (x > T->data)
    T->right = Delete(T->right, x);
  else if (x < T->data)
    T->left = Delete(T->left, x);
  else {
    if (T->right == NULL) {
      p = T;
      T = T->left;
      free(p);
    } else if (T->left == NULL) {
      p = T;
      T = T->right;
      free(p);
    } else {
      p = T->right;
      while (p->left != NULL)
        p = p->left;
      T->data = p->data;
      T->right = Delete(T->right, p->data);
    }
  }
  if (T != NULL) {
    T->ht = height(T);
    if (height(T->left) - height(T->right) == 2) {
      if (height(T->left->left) >= height(T->left->right))
        T = LL(T);
      else
        T = LR(T);
    } else if (height(T->right) - height(T->left) == 2) {
      if (height(T->right->left) >= height(T->right->right))
        T = RR(T);
      else
        T = RL(T);
    }
  }
  return T;
}

int height(node *T) {
  int lh, rh;
  if (T == NULL)
    return 0;
  if (T->left == NULL)
    lh = 0;
  else
    lh = 1 + height(T->left);
  if (T->right == NULL)
    rh = 0;
  else
    rh = 1 + height(T->right);
  return (lh > rh) ? lh : rh;
}

node *rotateleft(node *x) {
  node *y;
  y = x->right;
  x->right = y->left;
  y->left = x;
  x->ht = height(x);
  y->ht = height(y);
  return y;
}
```

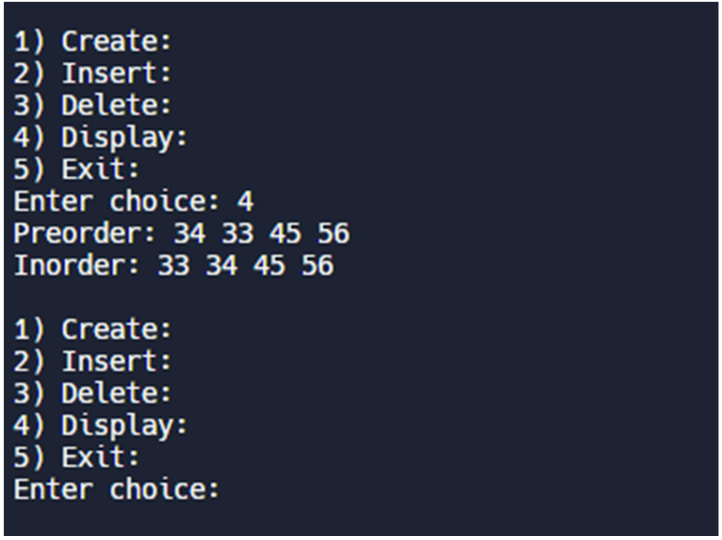
```
node *rotateright(node *x) {
  node *y;
  y = x->left;
  x->left = y->right;
  y->right = x;
  x->ht = height(x);
  y->ht = height(y);
  return y;
}

node *RR(node *T) {
  T = rotateleft(T);
  return T;
}

node *LL(node *T) {
  T = rotateright(T);
  return T;
}

node *LR(node *T) {
  T->left = rotateleft(T->left);
  T = rotateright(T);
  return T;
}

node *RL(node *T) {
  T->right = rotateright(T->right);
  T = rotateleft(T);
  return T;
}

void preorder(node *T) {
  if (T != NULL) {
    printf("%d ", T->data);
    preorder(T->left);
    preorder(T->right);
  }
}

void inorder(node *T) {
  if (T != NULL) {
    inorder(T->left);
    printf("%d ", T->data);
    inorder(T->right);
  }
}

int main() {
  node *root = NULL;
  int x, n, op;
  do {
    printf("\n1) Create: ");
    printf("\n2) Insert: ");
    printf("\n3) Delete: ");
    printf("\n4) Display: ");
    printf("\n5) Exit: ");
    printf("\nEnter choice: ");
    scanf("%d", &op);
    switch (op) {
```

```
  case 1:
    printf("Enter number of elements to insert: ");
    scanf("%d", &n);
    printf("Enter elements: ");
    for (int i = 0; i < n; i++) {
      scanf("%d", &x);
      root = insert(root, x);
    }
    break;
  case 2:
    printf("Enter element to insert: ");
    scanf("%d", &x);
    root = insert(root, x);
    break;
  case 3:
    printf("Enter element to delete: ");
    scanf("%d", &x);
    root = Delete(root, x);
    break;
  case 4:
    printf("Preorder: ");
    preorder(root);
    printf("\nInorder: ");
    inorder(root);
    printf("\n");
    break;
  }
} while (op != 5);
return 0;
```

```
}
```



```
1) Create:
2) Insert:
3) Delete:
4) Display:
5) Exit:
Enter choice: 1
Enter number of elements to insert: 3
Enter elements: 34
45
56

1) Create:
2) Insert:
3) Delete:
4) Display:
5) Exit:
Enter choice: 4
Preorder: 34 45 56
Inorder: 34 45 56

1) Create:
2) Insert:
3) Delete:
4) Display:
5) Exit:
Enter choice: 2
Enter element to insert: 33

1) Create:
2) Insert:
```



```
1) Create:
2) Insert:
3) Delete:
4) Display:
5) Exit:
Enter choice: 4
Preorder: 34 33 45 56
Inorder: 33 34 45 56

1) Create:
2) Insert:
3) Delete:
4) Display:
5) Exit:
Enter choice:
```