```c
#include <stdio.h>
typedef struct Node {
    int data;
    int color;
    struct Node *left, *right, *parent;
} Node;
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->color = RED;
    newNode->left = newNode->right = newNode->parent = NULL;
    return newNode;
}
void rotateLeft(Node** root, Node* x) {
    Node* y = x->right;
    x->right = y->left;
    if (y->left != NULL)
        y->left->parent = x;
    y->parent = x->parent;
    if (x->parent == NULL)
        *root = y;
    else if (x == x->parent->left)
        x->parent->left = y;
    else
        x->parent->right = y;
    y->left = x;
    x->parent = y;
}
void rotateRight(Node** root, Node* y) {
    Node* x = y->left;
    y->left = x->right;
    if (x->right != NULL)
        x->right->parent = y;
    x->parent = y->parent;
    if (y->parent == NULL)
        *root = x;
```

```
    else if (y == y->parent->left)
        y->parent->left = x;
    else
        y->parent->right = x;
    x->right = y;
    y->parent = x;
}
void fixViolation(Node** root, Node* node) {
    Node* parent = NULL;
    Node* grandparent = NULL;
    while (node != *root && node->parent->color == RED) {
        parent = node->parent;
        grandparent = parent->parent;
        if (parent == grandparent->left) {
            Node* uncle = grandparent->right;
            if (uncle != NULL && uncle->color == RED) {
                grandparent->color = RED;
                parent->color = BLACK;
                uncle->color = BLACK;
                node = grandparent;
            } else {
                if (node == parent->right) {
                    node = parent;
                    rotateLeft(root, node);
                }
                parent->color = BLACK;
                grandparent->color = RED;
                rotateRight(root, grandparent);
            }
        } else {
            Node* uncle = grandparent->left;
            if (uncle != NULL && uncle->color == RED) {
                grandparent->color = RED;
                parent->color = BLACK;
                uncle->color = BLACK;
                node = grandparent;
```

```
            } else {
                if (node == parent->left) {
                    node = parent;
                    rotateRight(root, node);
                }
                parent->color = BLACK;
                grandparent->color = RED;
                rotateLeft(root, grandparent);
            }
        }
    }
    (*root)->color = BLACK;
}
void insert(Node** root, int data) {
    Node* newNode = createNode(data);
    Node* y = NULL;
    Node* x = *root;
    while (x != NULL) {
        y = x;
        if (newNode->data < x->data)
            x = x->left;
        else
            x = x->right;
    }
    newNode->parent = y;
    if (y == NULL)
        *root = newNode;
    else if (newNode->data < y->data)
        y->left = newNode;
    else
        y->right = newNode;
    fixViolation(root, newNode);
}
void inorder(Node* root) {
    if (root != NULL) {
        inorder(root->left);
```

```
        inorder(root->left);
        printf("%d(%s) ", root->data, root->color == RED ? "R" : "B");
        inorder(root->right);
    }
}
int main() {
    Node* root = NULL;
    insert(&root, 10);
    insert(&root, 20);
    insert(&root, 30);
    insert(&root, 15);
    insert(&root, 25);
    printf("Inorder Traversal of the Red-Black Tree:\n");
    inorder(root);
    printf("\n");
    return 0;
}
```