```
/Created By Ritwik Chandra Pandey
//On 25 Oct 2021
//Implementation of RED-BLACK tree - rotations, insertion and inorder traversal


#include<stdio.h>
#include<conio.h>



typedef char COLOR;
struct node {
	int data;
	COLOR color;
	struct node *left, *right,*parent;
};
typedef struct node * RBNODE;
RBNODE root = NULL;

void leftRotate(RBNODE x) {
	RBNODE y;
	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->parent->left!=NULL && x->data == x->parent->left->data){
		x->parent->left=y;
	}else{
		x->parent->right = y;
	}
	y->left = x;
	x->parent = y;
	return;
}
```

```
void rightRotate(RBNODE y) {
	RBNODE x;
	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->parent->left!=NULL && y->data == y->parent->left->data){
		y->parent->left = x;
	}
	else{
		y->parent->right = x;
	}
	x->right = y;
	y->parent = x;
	return;
}

void colorInsert(RBNODE z) {
	RBNODE y=NULL;
	while(z->parent!=NULL && z->parent->color=='r'){
		if(z->parent->parent->left!=NULL && z->parent->data == z->parent->parent->left->data){
			if(z->parent->parent->right!=NULL){
				y = z->parent->parent->right;
			}
			if(y!=NULL&& y->color=='r'){
				z->parent->color='b';
				y->color='b';
				z->parent->parent->color = 'r';
				if(z->parent->parent!=NULL){
					z=z->parent->parent;
				}
			}
			else{
				if(z->parent->right!=NULL && z->data == z->parent->right->data){
```

```
                                        z=z->parent;
                                        leftRotate(z);
                                }
                                z->parent->color = 'b';
                                z->parent->parent->color = 'r';
                                rightRotate(z->parent->parent);
                        }
                }
                else{
                        if(z->parent->parent->left!=NULL){
                                y = z->parent->parent->left;
                        }
                        if(y!=NULL && y->color=='r'){
                                z->parent->color = 'b';
                                y->color='b';
                                z->parent->parent->color = 'r';
                                if(z->parent->parent!=NULL){
                                        z = z->parent->parent;
                                }
                        }
                        else{
                                if(z->parent->left!=NULL && z->data == z->parent->left->data){
                                        z = z->parent;
                                        rightRotate(z);
                                }
                                z->parent->color = 'b';
                                z->parent->parent->color = 'r';
                                leftRotate(z->parent->parent);
                        }
                }
        }
        root->color = 'b';
}

int searchNodeInRB(int val) {
        RBNODE temp = root;
        while(temp!=NULL){
                if(temp->data < val){
                        temp = temp->right;
                }else if(temp->data>val){
```

```
                        temp = temp->left;
                }else{
                        return 1;
                }
        }
        return 0;
}

void insertNodeInRB(int ele) {
        RBNODE x,y;
        RBNODE z = (RBNODE)malloc(sizeof(struct node));
        z->data = ele;
        z->left = NULL;
        z->right = NULL;
        z->parent = NULL;
        z->color = 'r';
        x=root;
        if(searchNodeInRB(ele)==1){
                printf("Entered element already exists in the RBTree.\n");
                return;
        }
        if(root==NULL){
                root = z;
                root->color = 'b';
                return;
        }
        while(x!=NULL){
                y=x;
                if(z->data<x->data){
                        x = x->left;
                }else{
                        x = x->right;
                }
        }
        z->parent = y;
        if(y==NULL){
                root=z;
        }else if(z->data<y->data){
                y->left = z;
        }else{
```

```
                y->right = z;
        }
        colorInsert(z);
        return;
}

void inorderInRB(RBNODE root) {
        if(root!=NULL){
                inorderInRB(root->left);
                        printf("%d(%c)  ", root->data,root->color);
                inorderInRB(root->right);
        }
}


void main() {
        int ele, op;
        while(1)
        {
                printf("1.Insert 2.Inorder Traversal 3.Exit\n");
                printf("Enter your option : ");
                scanf("%d", &op);
                switch(op) {
                        case 1:printf("Enter an element to be inserted : ");
                                        scanf("%d", &ele);
                                        insertNodeInRB(ele);
                                        break;
                        case 2:
                                        if(root == NULL) {
                                                printf("RBTree is empty.\n");
                                        }
                                        else {
                                                printf("Elements of the RB tree (in-order traversal): ");
                                                inorderInRB(root);
                                                printf("\n");
                                        }
                                        break;
                        case 3:
                                        exit(0);
                }
```

```
        }
}
```