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

typedef enum { RED, BLACK } node_color;

typedef struct RBNode {
    int key;
    node_color color;
    struct RBNode *left, *right, *parent;
} RBNode;

/* The NIL sentinel node. All leaves point to this single sentinel.
    */
RBNode *NIL;
RBNode *root;
```

```c
/* Delete node with key (first occurrence) */
void rb_delete(int key) {
    RBNode *z = root;
    while (z != NIL && z->key != key) {
        if (key < z->key) z = z->left;
        else z = z->right;
    }
    if (z == NIL) {
        printf("Key %d not found, nothing deleted \n", key);
```

```c
RBNode* create_node(int key) {
    RBNode *node = (RBNode*)malloc(sizeof(RBNode));
    if (!node) {
        fprintf(stderr, "Memory allocation failed\n");
        exit(EXIT_FAILURE);
    }
    node->key = key;
    node->color = RED;
    node->left = node->right = node->parent = NIL;
    return node;
}

/* Left rotate x */
void left_rotate(RBNode *x) {
    RBNode *y = x->right;
    x->right = y->left;
    if (y->left != NIL) y->left->parent = x;
    y->parent = x->parent;
    if (x->parent == NIL) root = y;
    else if (x == x->parent->left) x->parent->left = y;
    else x->parent->right = y;
    y->left = x;
    x->parent = y;
}

/* Right rotate x */
```