NAME: JERRY DAVID R (192424401)

COURSE NAME: DATA STRUCTURES FOR MODERN COMPUTING SYSTEMS

COURSE CODE: CSA0302

Experiment 27: Splay Tree

Code:

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

struct Node {
    int key;
    struct Node *left, *right;
};

// Function to create new node
struct Node* newNode(int key) {
    struct Node* node = (struct Node*)malloc(sizeof(struct Node));
    node->key = key;
    node->left = node->right = NULL;
    return node;
}

// Right rotate
struct Node* rightRotate(struct Node* x) {
    struct Node* y = x->left;
    x->left = y->right;
    y->right = x;
    return y;
}
```

```
// Left rotate
struct Node* leftRotate(struct Node* x) {
    struct Node* y = x->right;
    x->right = y->left;
    y->left = x;
    return y;
}

// Splay operation
struct Node* splay(struct Node* root, int key) {
    if (root == NULL || root->key == key)
        return root;

    // Key lies in left subtree
    if (key < root->key) {
        if (root->left == NULL) return root;

        // Zig-Zig (Left Left)
        if (key < root->left->key) {
            root->left->left = splay(root->left->left, key);
            root = rightRotate(root);
        }
        // Zig-Zag (Left Right)
        else if (key > root->left->key) {
            root->left->right = splay(root->left->right, key);
            if (root->left->right != NULL)
                root->left = leftRotate(root->left);
        }

        return (root->left == NULL) ? root : rightRotate(root);
    }
```

```
    // Key lies in right subtree
    else {
        if (root->right == NULL) return root;

        // Zag-Zig (Right Left)
        if (key < root->right->key) {
            root->right->left = splay(root->right->left, key);
            if (root->right->left != NULL)
                root->right = rightRotate(root->right);
        }
        // Zag-Zag (Right Right)
        else if (key > root->right->key) {
            root->right->right = splay(root->right->right, key);
            root = leftRotate(root);
        }

        return (root->right == NULL) ? root : leftRotate(root);
    }
}

// Insert operation
struct Node* insert(struct Node* root, int key) {
    if (root == NULL) return newNode(key);

    root = splay(root, key);

    if (root->key == key) return root;

    struct Node* node = newNode(key);
```

```
    if (key < root->key) {
        node->right = root;
        node->left = root->left;
        root->left = NULL;
    } else {
        node->left = root;
        node->right = root->right;
        root->right = NULL;
    }

    return node;
}

// Preorder traversal
void preorder(struct Node* root) {
    if (root != NULL) {
        printf("%d ", root->key);
        preorder(root->left);
        preorder(root->right);
    }
}

// Menu-driven main function
int main() {
    struct Node* root = NULL;
    int choice, key;

    while (1) {
        printf("\n--- Splay Tree Menu ---\n");
        printf("1. Insert\n2. Splay\n3. Display (Preorder)\n4. Exit\n");
        printf("Enter your choice: ");
```

```
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to insert: ");
                scanf("%d", &key);
                root = insert(root, key);
                printf("Value inserted and splayed to root.\n");
                break;
            case 2:
                printf("Enter value to search: ");
                scanf("%d", &key);
                root = splay(root, key);
                if (root && root->key == key)
                    printf("Element %d found and splayed to root.\n", key);
                else
                    printf("Element %d not found.\n", key);
                break;
            case 3:
                printf("Preorder traversal: ");
                preorder(root);
                printf("\n");
                break;
            case 4:
                exit(0);
            default:
                printf("Invalid choice!\n");
        }
    }
}
```

Output:

```
--- Splay Tree Menu ---
1. Insert
2. Splay
3. Display (Preorder)
4. Exit
Enter your choice: 1
Enter value to insert: 10
Value inserted and splayed to root.

--- Splay Tree Menu ---
1. Insert
2. Splay
3. Display (Preorder)
4. Exit
Enter your choice: 1
Enter value to insert: 0
Value inserted and splayed to root.

--- Splay Tree Menu ---
1. Insert
2. Splay
3. Display (Preorder)
4. Exit
Enter your choice: 1
Enter value to insert: 15
Value inserted and splayed to root.

--- Splay Tree Menu ---
1. Insert
2. Splay
3. Display (Preorder)
4. Exit
Enter your choice: 2
Enter value to search: 15
Element 15 found and splayed to root.

--- Splay Tree Menu ---
1. Insert
2. Splay
3. Display (Preorder)
4. Exit
Enter your choice: 3
Preorder traversal: 15 10 0
```