Week 8

Doubly Linked List

#include <stdio.h>

#include <stdlib.h>

typedef struct Node {

int data;

struct Node\* prev;

struct Node\* next;

} Node;

Node\* createNode(int data) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->data = data;

newNode->prev = NULL;

newNode->next = NULL;

return newNode;

}

Node\* createList(Node\* head, int data) {

Node\* newNode = createNode(data);

if (head == NULL) {

head = newNode;

} else {

Node\* temp = head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = newNode;

newNode->prev = temp;

}

return head;

}

Node\* insertLeft(Node\* head, int target, int data) {

Node\* temp = head;

while (temp != NULL && temp->data != target) {

temp = temp->next;

}

if (temp == NULL) return head;

Node\* newNode = createNode(data);

newNode->next = temp;

newNode->prev = temp->prev;

if (temp->prev != NULL) temp->prev->next = newNode;

else head = newNode;

temp->prev = newNode;

return head;

}

Node\* deleteNode(Node\* head, int value) {

Node\* temp = head;

while (temp != NULL && temp->data != value) {

temp = temp->next;

}

if (temp == NULL) return head;

if (temp->prev != NULL) temp->prev->next = temp->next;

else head = temp->next;

if (temp->next != NULL) temp->next->prev = temp->prev;

free(temp);

return head;

}

void displayList(Node\* head) {

Node\* temp = head;

while (temp != NULL) {

printf("%d ", temp->data);

temp = temp->next;

}

printf("\n");

}

int main() {

Node\* head = NULL;

int choice, data, target;

printf("Ruqaiyya Mahreen 1BM23EE044\n");

do {

printf("1.Create List 2.Insert Left 3.Delete Node 4.Display List 5.Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter data: ");

scanf("%d", &data);

head = createList(head, data);

break;

case 2:

printf("Enter target node value and new data: ");

scanf("%d %d", &target, &data);

head = insertLeft(head, target, data);

break;

case 3:

printf("Enter value to delete: ");

scanf("%d", &data);

head = deleteNode(head, data);

break;

case 4:

displayList(head);

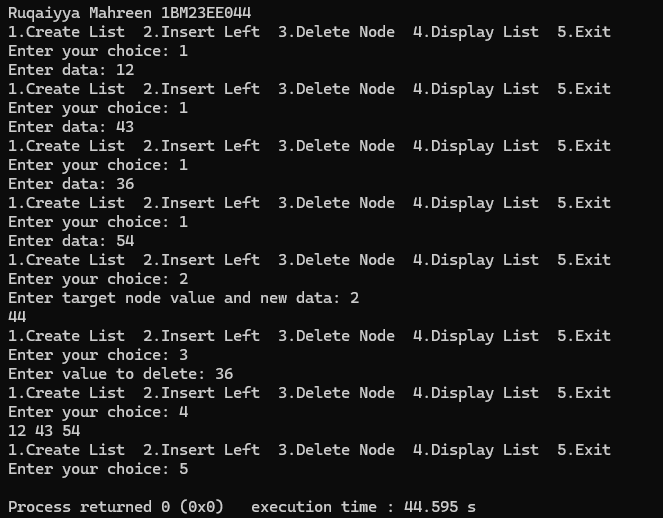
break;

}

} while (choice != 5);

return 0;

}



Trees

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

struct Node\* createNode(int data);

struct Node\* insertNode(struct Node\* root, int data);

void inorderTraversal(struct Node\* root);

void preorderTraversal(struct Node\* root);

void postorderTraversal(struct Node\* root);

void displayTree(struct Node\* root);

struct Node\* createNode(int data) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = data;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

struct Node\* insertNode(struct Node\* root, int data) {

if (root == NULL) {

return createNode(data);

}

if (data < root->data) {

root->left = insertNode(root->left, data);

} else if (data > root->data) {

root->right = insertNode(root->right, data);

}

return root;

}

void inorderTraversal(struct Node\* root) {

if (root != NULL) {

inorderTraversal(root->left);

printf("%d ", root->data);

inorderTraversal(root->right);

}

}

void preorderTraversal(struct Node\* root) {

if (root != NULL) {

printf("%d ", root->data);

preorderTraversal(root->left);

preorderTraversal(root->right);

}

}

void postorderTraversal(struct Node\* root) {

if (root != NULL) {

postorderTraversal(root->left);

postorderTraversal(root->right);

printf("%d ", root->data);

}

}

void displayTree(struct Node\* root) {

printf("Tree elements : ");

inorderTraversal(root);

printf("\n");

}

int main() {

struct Node\* root = NULL;

int choice, data;

printf("Ruqaiyya Mahreen 1BM23EE044\n");

while (1) {

printf("1.Insert an element into BST 2.Inorder traversal 3.Preorder traversal 4.Postorder traversal 5.Display all elements 6.Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter the element to insert: ");

scanf("%d", &data);

root = insertNode(root, data);

break;

case 2:

printf("Inorder Traversal: ");

inorderTraversal(root);

printf("\n");

break;

case 3:

printf("Preorder Traversal: ");

preorderTraversal(root);

printf("\n");

break;

case 4:

printf("Postorder Traversal: ");

postorderTraversal(root);

printf("\n");

break;

case 5:

displayTree(root);

break;

case 6:

printf("Exiting program.\n");

exit(0);

default:

printf("Invalid choice! Please try again.\n");

}

}

return 0;

}

