**DAA Assignment-2**

**1)Programs for summation of series 1+X+X^2+X^3+…with different time**

**complexities.**

**ANS :** **a)Using Iterative Approach**

#include <stdio.h>

#include<math.h>

int main() {

int n, i;

int X, sum = 0, term = 1;

printf("Enter the value of X: ");

scanf("%d", &X);

printf("Enter the value of n: ");

scanf("%d", &n);

for(i = 0; i <= n; i++) {

sum = sum+term;

term = term\*X;

}

printf("Sum of the series: %d", sum);

return 0;

}

A screenshot of a computer

AI-generated content may be incorrect.

The Time Complexity for this method is O(n) because we used a FOR Loop in it.

**b)Using Geometric Series Formula**

#include <stdio.h>

#include <math.h>

int main() {

int n;

int X, sum;

printf("Enter the value of X: ");

scanf("%d", &X);

printf("Enter the value of n: ");

scanf("%d", &n);

if(X != 1) {

sum = (pow(X, n + 1) - 1) / (X - 1);

} else {

sum = n + 1;

}

printf("Sum of the series: %d\n", sum);

return 0;

}

A screenshot of a computer

AI-generated content may be incorrect.

The Time Complexity for this methodO(1) ie. Constant .

**2) Create a Binary Search Tree and perform the insertion, deletion operations.**

**ANS :** #include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

struct Node\* createNode(int data) {

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

newNode->data = data;

newNode->left = newNode->right = NULL;

return newNode;

}

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

if (root == NULL) {

return createNode(data);

}

if (data < root->data) {

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

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

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

}

return root;

}

struct Node\* findMin(struct Node\* node) {

struct Node\* current = node;

while (current && current->left != NULL) {

current = current->left;

}

return current;

}

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

if (root == NULL) {

return root;

}

if (data < root->data) {

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

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

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

} else {

if (root->left == NULL) {

struct Node\* temp = root->right;

free(root);

return temp;

} else if (root->right == NULL) {

struct Node\* temp = root->left;

free(root);

return temp;

}

struct Node\* temp = findMin(root->right);

root->data = temp->data;

root->right = deleteNode(root->right, temp->data);

}

return root;

}

void inorderTraversal(struct Node\* root) {

if (root != NULL) {

inorderTraversal(root->left);

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

inorderTraversal(root->right);

}

}

int main() {

struct Node\* root = NULL;

root = insert(root, 50);

insert(root, 30);

insert(root, 20);

insert(root, 40);

insert(root, 70);

insert(root, 60);

insert(root, 80);

printf("Inorder traversal of the created BST: \n");

inorderTraversal(root);

printf("\n");

printf("\nDeleting 20...\n");

root = deleteNode(root, 20);

printf("Inorder traversal after deleting 20: \n");

inorderTraversal(root);

printf("\n");

printf("\nDeleting 30...\n");

root = deleteNode(root, 30);

printf("Inorder traversal after deleting 30: \n");

inorderTraversal(root);

printf("\n");

printf("\nDeleting 50 (root node)...\n");

root = deleteNode(root, 50);

printf("Inorder traversal after deleting 50: \n");

inorderTraversal(root);

printf("\n");

return 0;

}

A screenshot of a computer

AI-generated content may be incorrect.