1. Method 1

#include <stdio.h>

#include <math.h>

double Series(double a, double r, int n) {

if (r == 1) {

return a \* n;

}

return (a \* (pow(r, n) - 1)) / (r - 1);

}

int main() {

double a , r;

int n;

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

scanf("%lf", &a);

printf(" the value of X: ");

scanf("%lf", &r);

printf("Enter number of terms : ");

scanf("%d", &n);

double sum = Series(a, r, n);

printf("Sum of the series = %.2lf\n", sum);

return 0;

}

Enter the value of a: 8

the value of X: 5

Enter number of terms : 8

Sum of the series = 781248.00

A screenshot of a computer

AI-generated content may be incorrect.

2nd method :

#include <stdio.h>

double Series(double a, double r, int n) {

double sum = 0.0;

for (int i = 1; i <= n; i++) {

sum = sum + a;

a = a \* r;

}

return sum;

}

int main() {

double a , r;

int n;

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

scanf("%lf", &a);

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

scanf("%lf", &r);

printf("Enter the number of terms: ");

scanf("%d", &n);

double sum = Series(a, r, n);

printf("Sum = %.2lf\n", sum);

return 0;

}

Enter the value of a: 5

Enter the value of x : 7

Enter the number of terms : 8

Sum of the series = 4804000.00

A screenshot of a computer

AI-generated content may be incorrect.

2 ques code

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

struct Node\* createNode(int value) {

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

newNode->data = value;

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

return newNode;

}

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

if (root == NULL) {

return createNode(value);

}

if (value < root->data) {

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

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

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

}

return root;

}

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

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

node = node->left;

}

return node;

}

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

if (root == NULL) return root;

if (value < root->data) {

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

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

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

} 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 inorder(struct Node\* root) {

if (root != NULL) {

inorder(root->left);

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

inorder(root->right);

}

}

int main() {

struct Node\* root = NULL;

int choice, value;

printf("\nBinary Search Tree Operations:\n");

printf("1. Insert\n");

printf("2. Delete\n");

printf("3. Display (In-order)\n");

printf("4. Exit\n");

while(1){

printf(" your choice: ");

scanf("%d", &choice);

switch(choice) {

case 1:

printf(" value to insert: ");

scanf("%d", &value);

root = insert(root, value);

break;

case 2:

printf(" value to delete: ");

scanf("%d", &value);

root = deleteNode(root, value);

break;

case 3:

printf("In-order Traversal: ");

inorder(root);

printf("\n");

break;

case 4:

printf("Exiting program.\n");

exit(0);

break;

default:

printf("Invalid choice!\n");

}

} while(choice != 4);

return 0;

}

Output :

Binary Search Tree Operations:

1. Insert

2. Delete

3. Display (In-order)

4. Exit

Enter your choice: 1

Enter value to insert: 76

Enter your choice: 1

Enter value to insert: 90

Enter your choice: 1

Enter value to insert: 65

Enter your choice: 1

Enter value to insert: 54

Enter your choice: 2

Enter value to delete: 65

Enter your choice: 3

In-order Traversal: 54 76 90

Enter your choice: 4

Exiting program.

A screenshot of a computer

AI-generated content may be incorrect.