LAB TASK – 2

1. Program for summation fo series 1+X+X^2+X^3+… with different time complexities

Method 1 :

Code :

#include<stdio.h>

#include<math.h>

int main(){

int a,n,r;

printf("Enter the first element \n");

scanf("%d",&a);

printf("Enter the common ratio value \n");

scanf("%d",&r);

printf("Enter the number of terms in the series \n");

scanf("%d",&n);

int sum = 0;

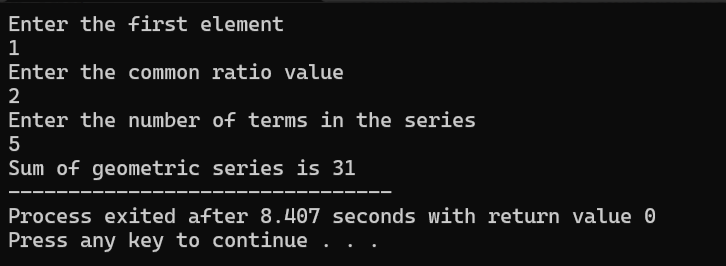
sum=(a\*pow(r,n)-1)/(r-1);

printf("Sum of geometric series is %d",sum);

return 0;

}

Output :



Method 2 :

Code :

#include<stdio.h>

int main(){

int a,n,r,i;

printf("Enter the first element \n");

scanf("%d",&a);

printf("Enter the common ratio value \n");

scanf("%d",&r);

printf("Enter the number of terms in the series \n");

scanf("%d",&n);

int sum = 0;

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

sum=sum+a;

a=a\*r;

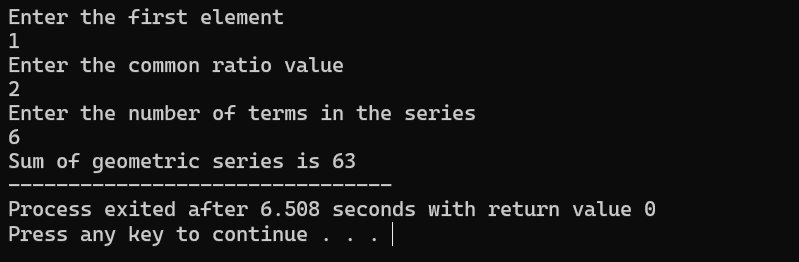
}

printf("Sum of geometric series is %d",sum);

return 0;

}

Output :



1. Create a binary Search Tree and perform insertion and deletion operations

Code :

#include <stdio.h>

#include <stdlib.h>

struct node {

int data;

struct node \*left, \*right;

};

struct node\* create(int value)

{

struct node\* temp= (struct node\*)malloc(sizeof(struct node));

temp->data = value;

temp->left = NULL;

temp->right = NULL;

return temp;

}

struct node\* insert(struct node\* root, int value)

{

if (root == NULL) {

return create(value);

}

if (value < root->data) {

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

}

else if (value > root->data) {

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

}

return root;

}

void inOrder(struct node\* root)

{

if (root != NULL) {

inOrder(root->left);

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

inOrder(root->right);

}

}

struct node\* findMin(struct node\* root)// Function to find the minimum value

{

if (root == NULL) {

return NULL;

}

else if (root->left != NULL) {

return findMin(root->left);

}

return root;

}

struct node\* delete (struct node\* root, int x)

{

if (root == NULL)

return NULL;

if (x > root->data) {

root->right = delete (root->right, x);

}

else if (x < root->data) {

root->left = delete (root->left, x);

}

else {

if (root->left == NULL && root->right == NULL) {

free(root);

return NULL;

}

else if (root->left == NULL || root->right == NULL) {

struct node\* temp;

if (root->left == NULL) {

temp = root->right;

}

else {

temp = root->left;

}

free(root);

return temp;

}

else {

struct node\* temp

= findMin(root->right);

root->data = temp->data;

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

}

}

return root;

}

int main()

{

struct node\* root = NULL;

int a[10],n,i,del1;

printf("Enter 10 numbers to insert into the binary search tree \n");

for ( i=0;i<10;i++){

scanf("%d",&a[i]);

}

for (i=0;i<10;i++){

root = insert(root,a[i]);

}

printf("Inorder Traversal Before Deletion \n");

inOrder(root);

printf("\n");

printf("Enter a number to be deleted \n");

scanf("%d",&del1);

root = delete (root,del1);

printf("Inorder Traversal After Deletion: \n");

inOrder(root);

return 0;

}

Output :

