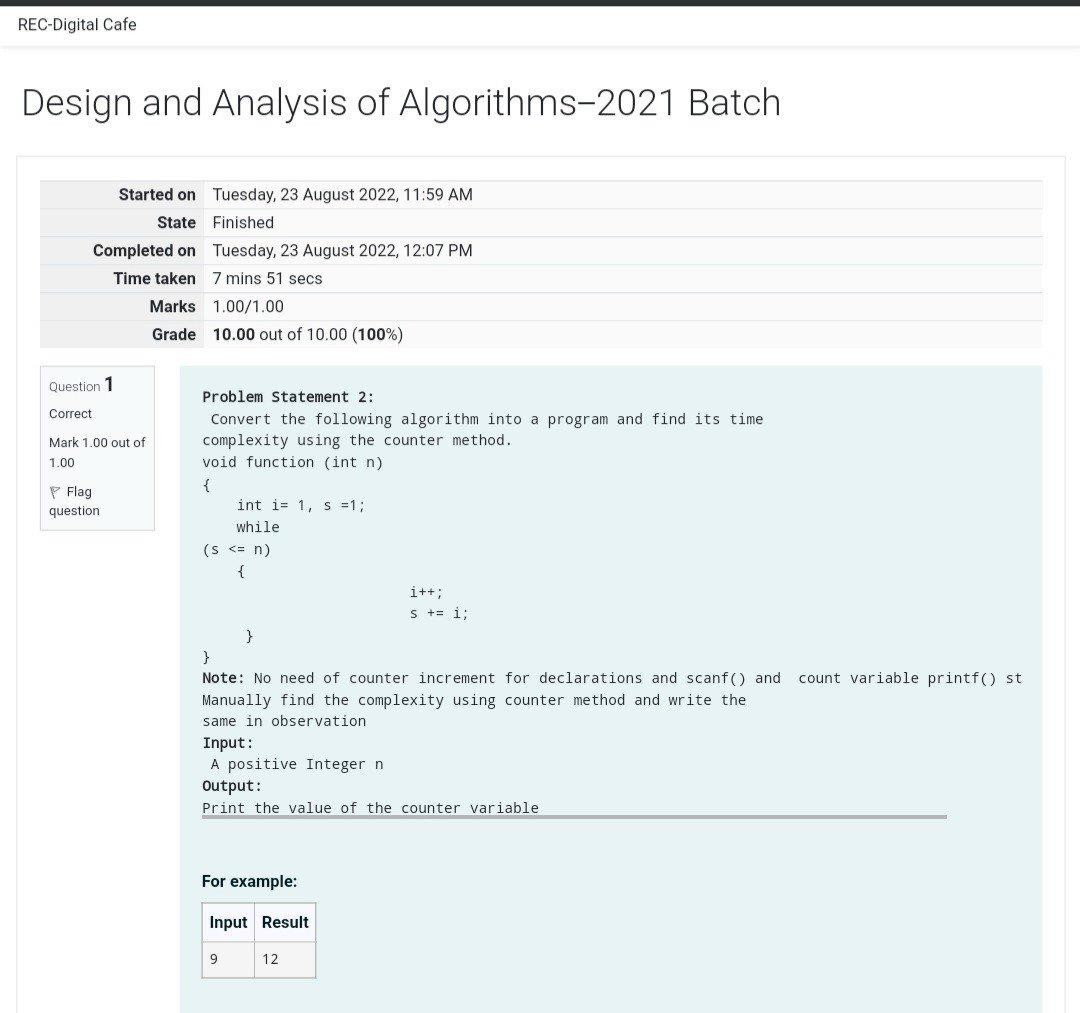
DAY-1:

CSA0672-DESIGN AND ANALYSIS OF ALGORITHM

NAME : K.Santhoshini

REGNO:192111121

1)



PROGRAM:

#include <stdio.h>

void function(int min);

int main()

{

int n;

printf(“enter the number:”);

scanf("%d",&n);

function(n);

return 0;

}

void function(int n)

{

int count=0;

int i=1,s=1;

count++;

count++;

while(s<=n)

{

count++;

i++;

count++;

s+=i;

count++;

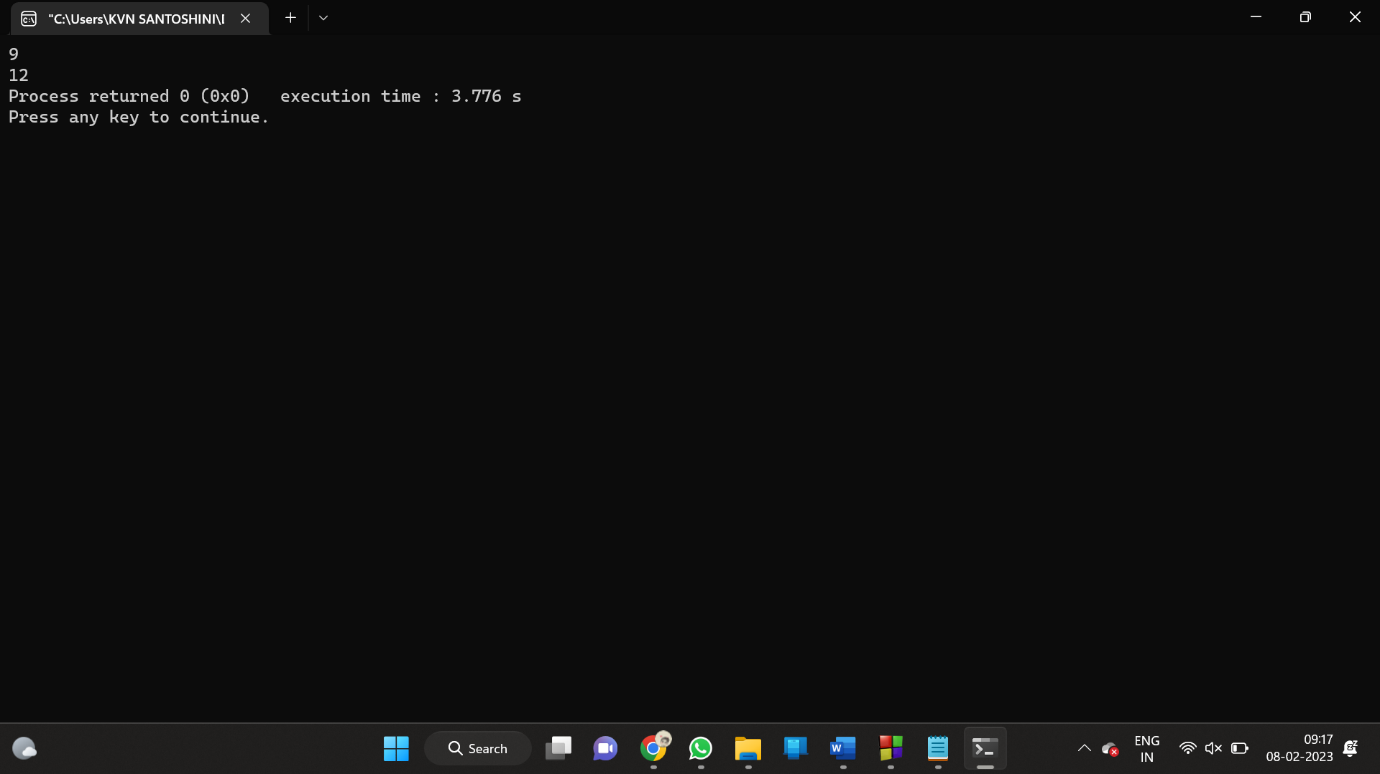
}

count++;

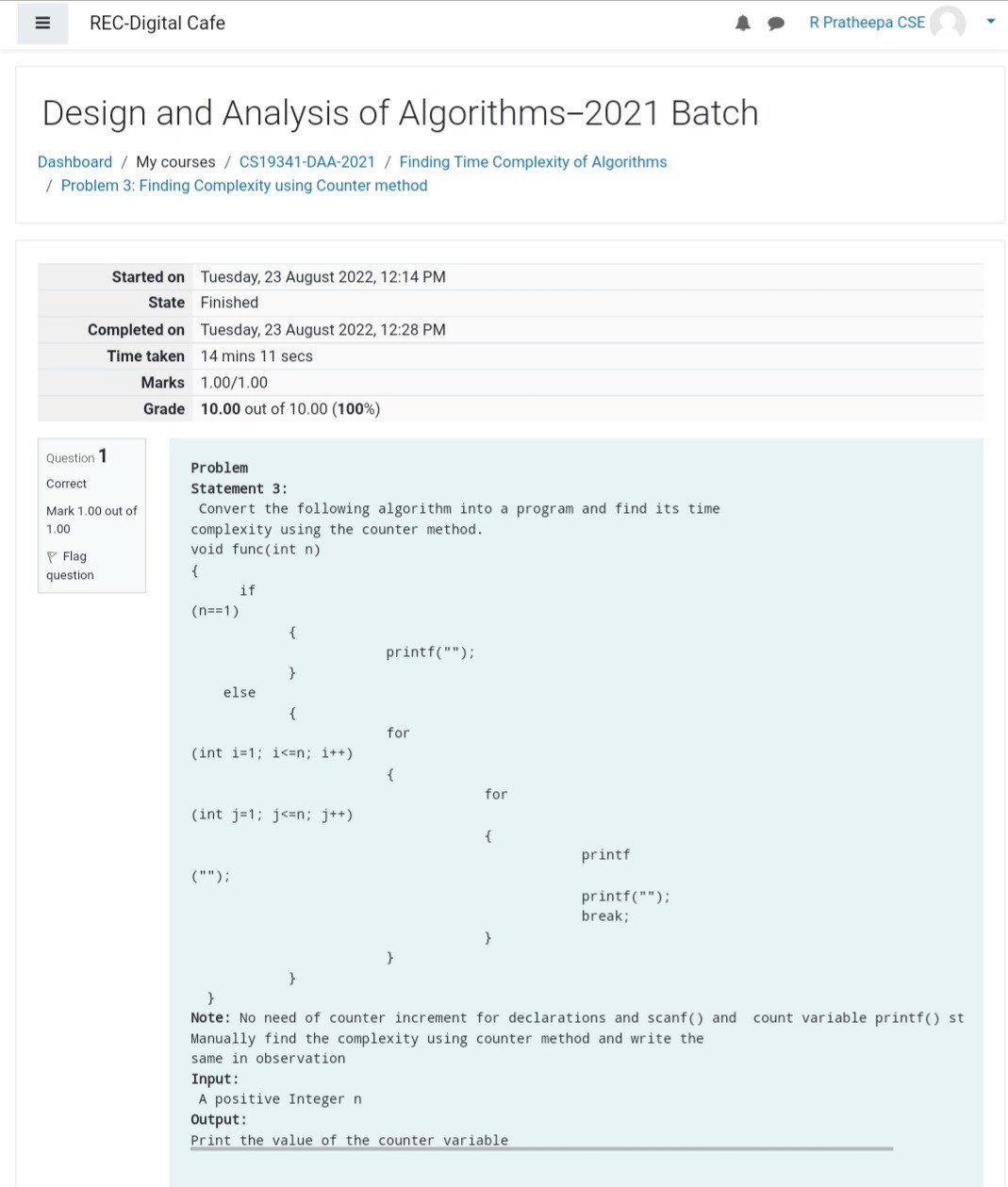
printf("%d",count);

}

OUTPUT:



2)



PROGRAM:

#include <stdio.h>

void function(int n);

int main()

{

int n;

scanf("%d",&n);

function(n);

return 0;

}

void function(int n)

{

int count=0;

if(n==1)

{

count++;

count++;

}

else

{

count++;

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

{

count++;

for(int j=1;j<=n;j++)

{

count++;

count++;

count++;

count++;

break;

}

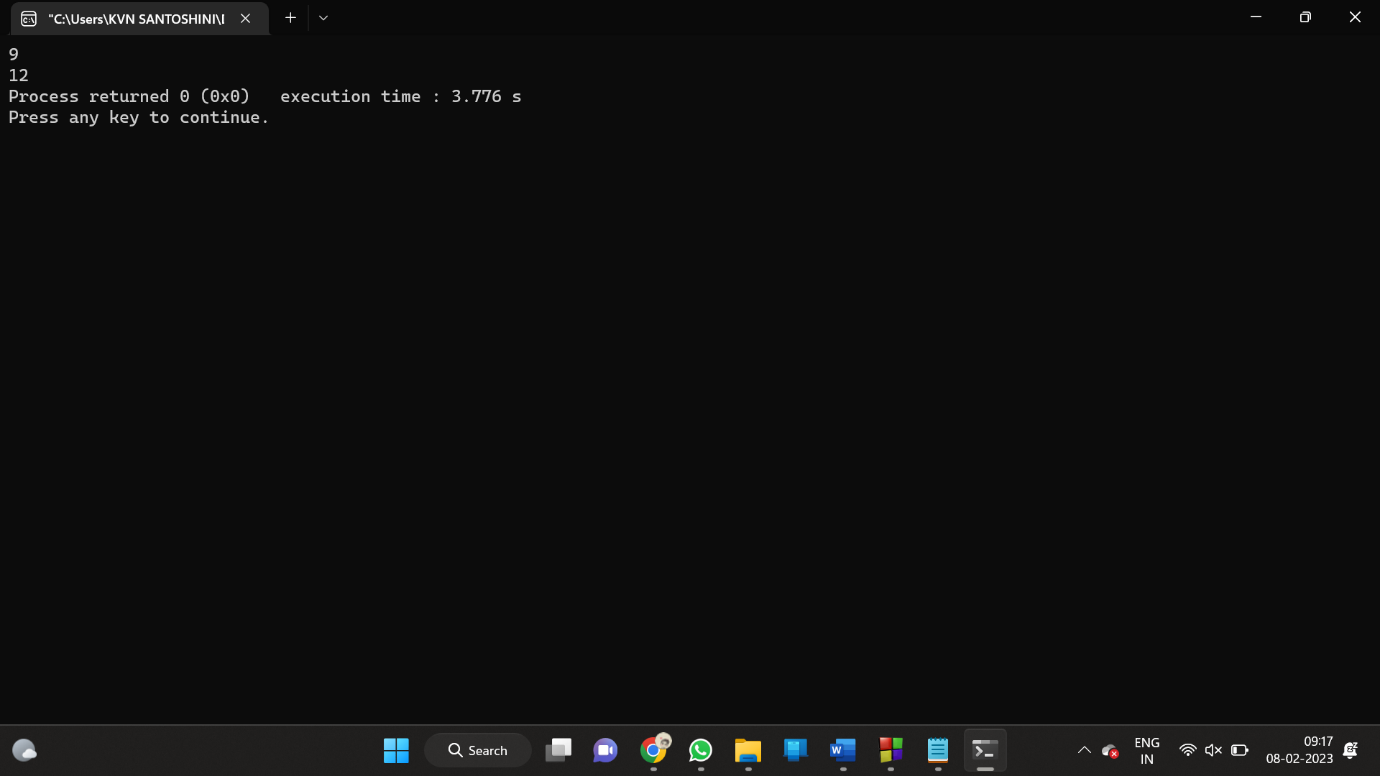
}count++;

}

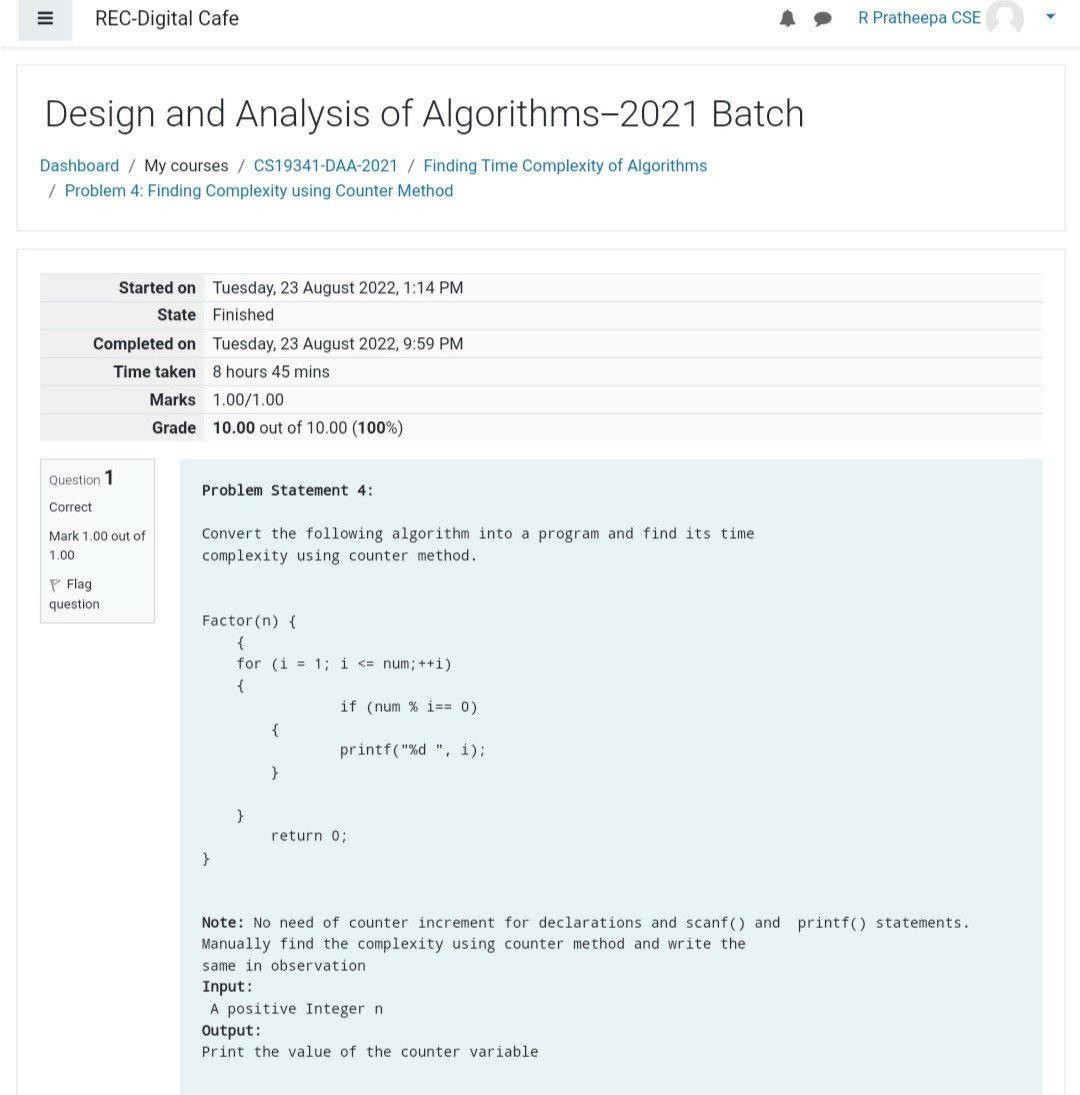
printf("%d",count);

}

OUTPUT:



3)



PROGRAM:

#include <stdio.h>

int factor(int n);

int main()

{

int n;

scanf("%d",&n);

factor(n);

return 0;

}

int factor( int n)

{

{

int i;

int count=0;

count++;

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

{

count++;

if(n%i==0)

{

count++;

printf("%d",i);

count++;

}

}

count++;

return 0;

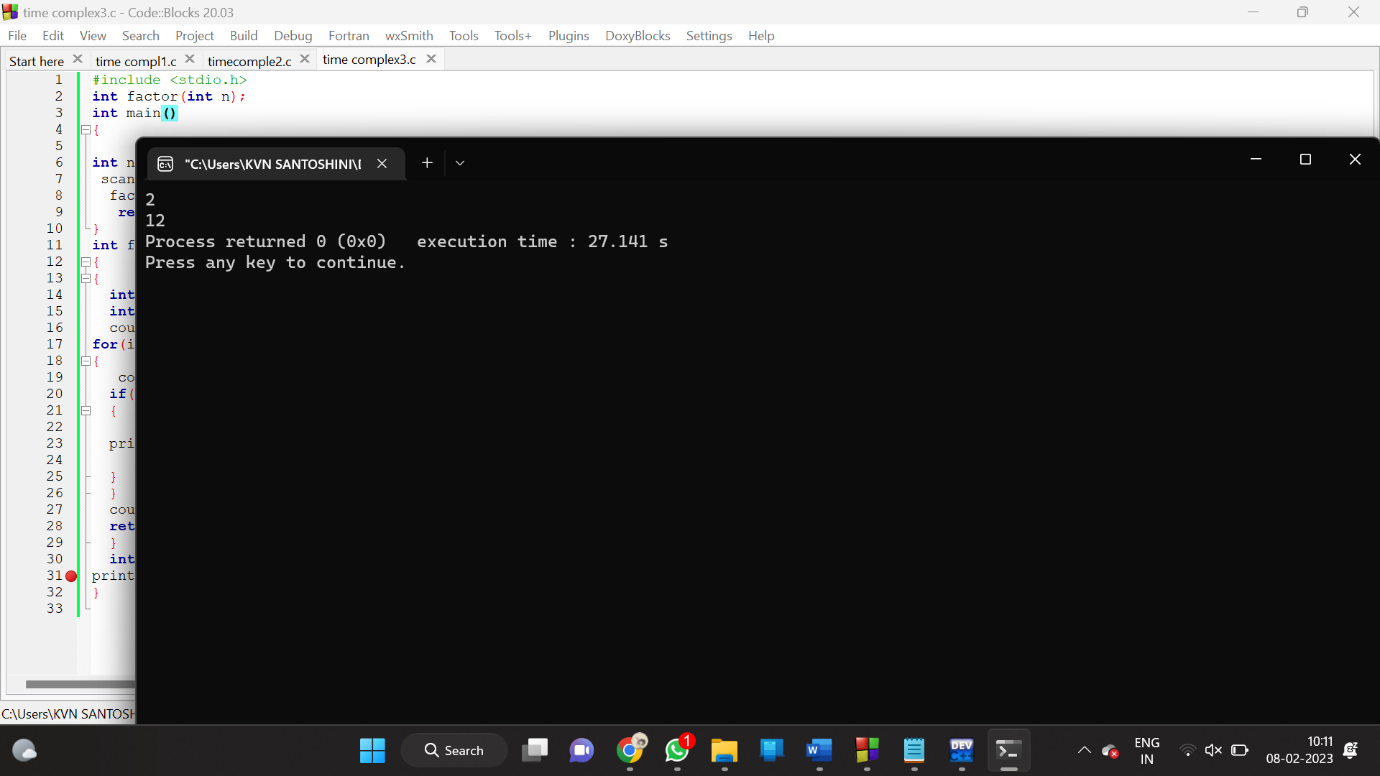
}

int count;

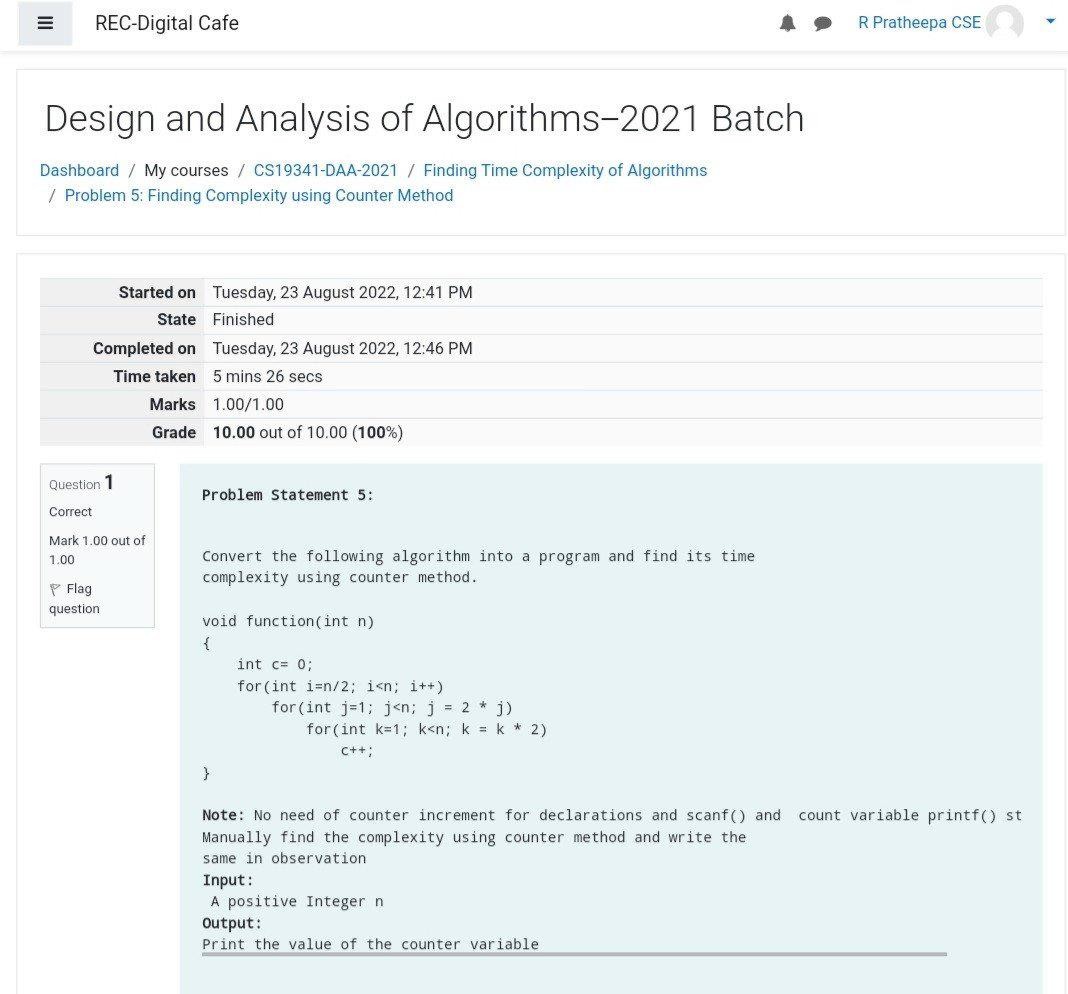
printf("%d",count);

}

OUTPUT:



4)



PROGRAM:

#include<stdio.h>

void function(int n);

int main()

{

int n;

scanf("%d",&n);

function(n);

return 0;

}

void function(int n)

{

int c=0;

int count=0;

count++;

for(int i=n/2;i<n;i++)

{

count++;

for(int j=1;j<n;j=2\*j)

{

count++;

for(int k=1; k<n;k=k\*2)

{

count++;

c++;

count++;

}count++;

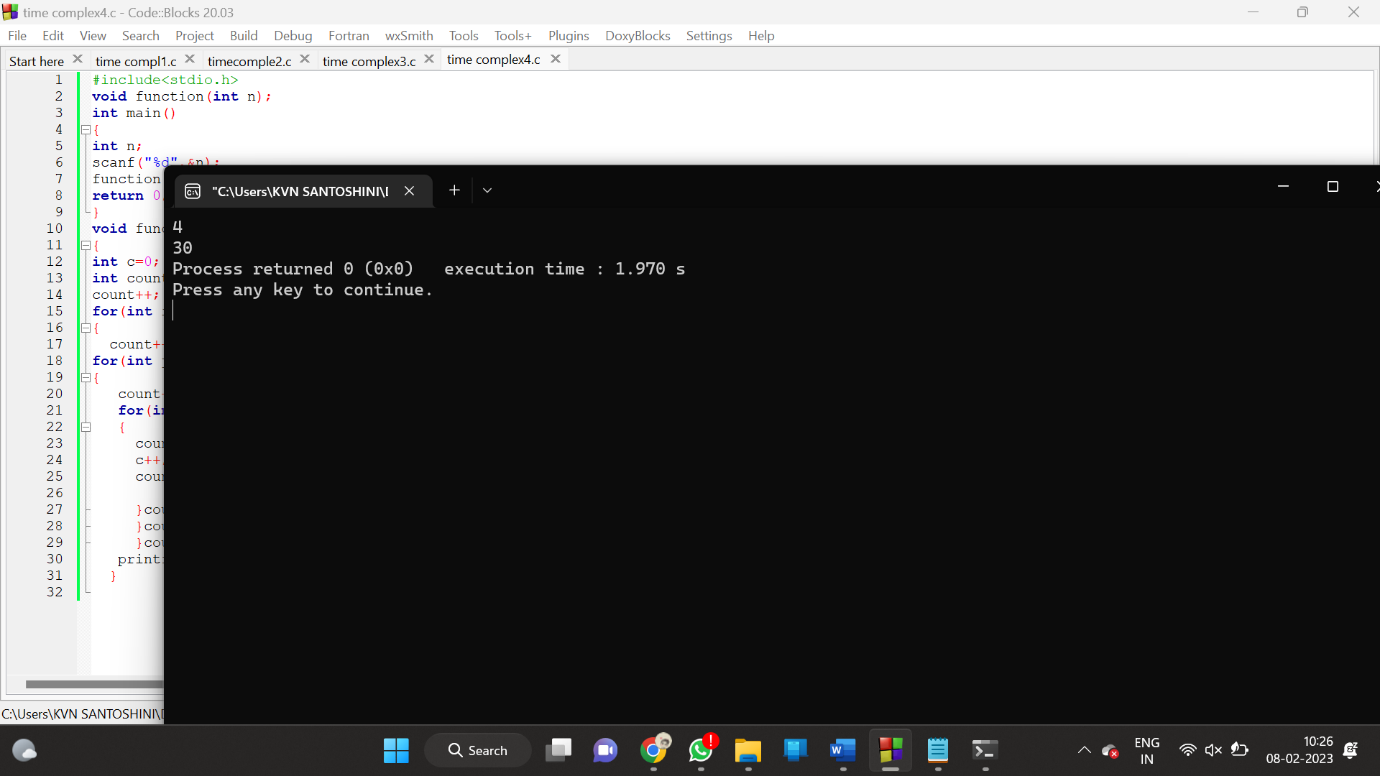
}count++;

}count++;

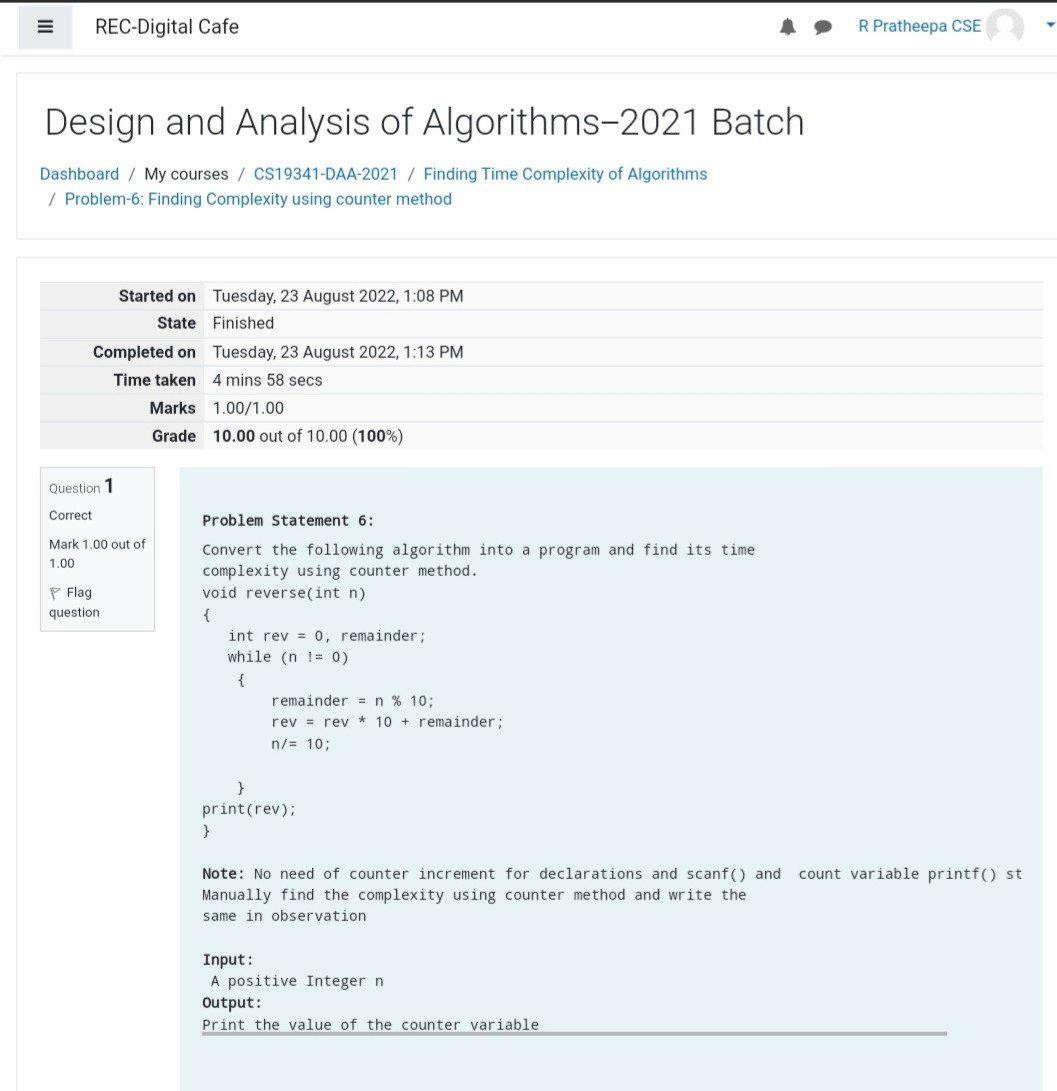
printf("%d",count);

}

OUTPUT:



5)



PROGRAM:

#include<stdio.h>

void reverse(int n);

int main()

{

int n;

scanf("%d",&n);

reverse(n);

return 0;

}

void reverse(int n)

{

int count=0;

int rev=0,remainder;

count++;

while(n!=0)

{

count++;

remainder=n%10;

count++;

rev=rev\*10+remainder;

count++;

n=n/10;

count++;

}

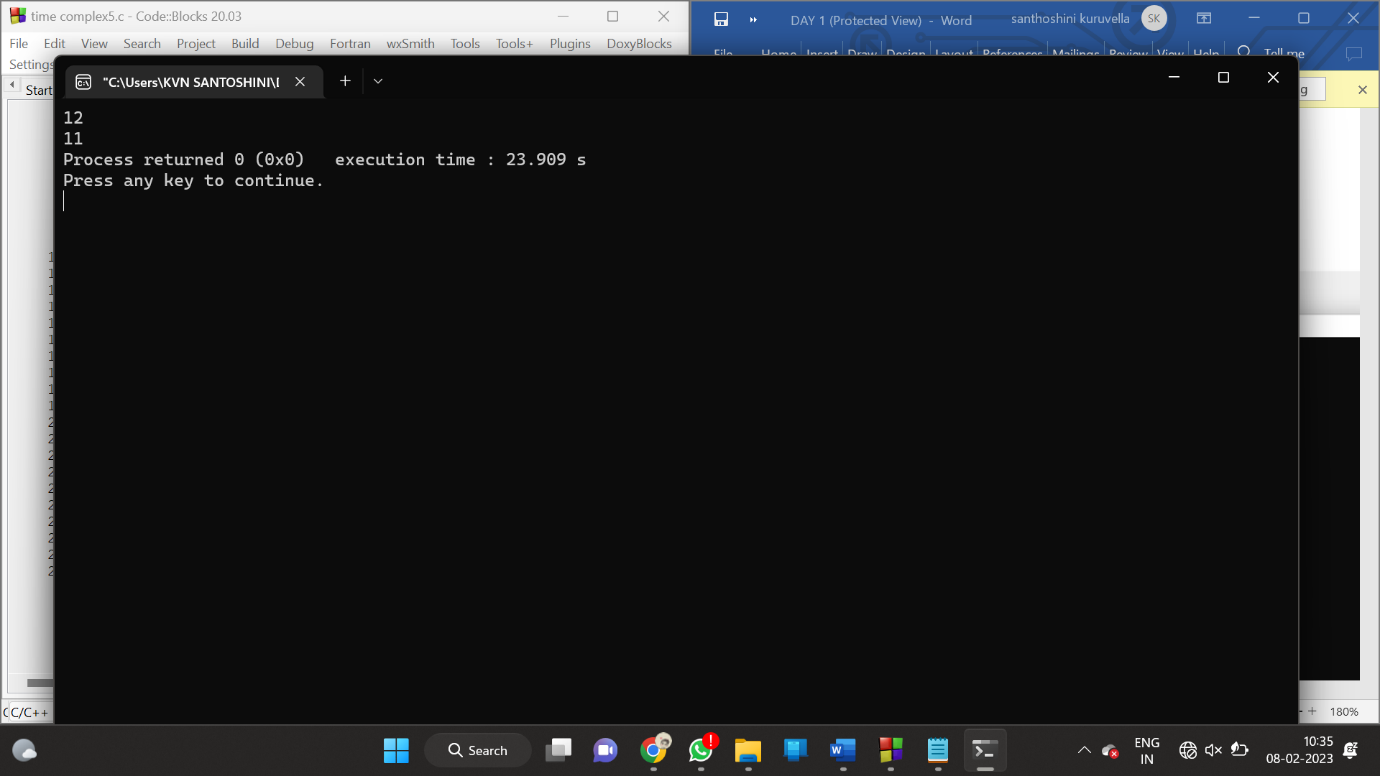
count++;

count++;

printf("%d",count);

}

OUTPUT:



6)PASCAL TRIANGLE FOR ROWS=5-TIME COMPLEXITY

#include<stdio.h>

int main()

{

int rows, coef = 1, space, i, j;

int count=0;

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

scanf("%d", &rows);

for (i = 0; i<rows; i++)

{

count++;

for (space = 1; space <= rows - i; space++)

printf(" ");

count++;

for (j = 0; j <= i; j++)

{

count++;

if(j == 0 || i == 0){

coef = 1;

count++;

}

else

{

coef = coef \* (i - j + 1) / j;

}

count++;

printf("%4d", coef);

}

printf("\n");

count++;

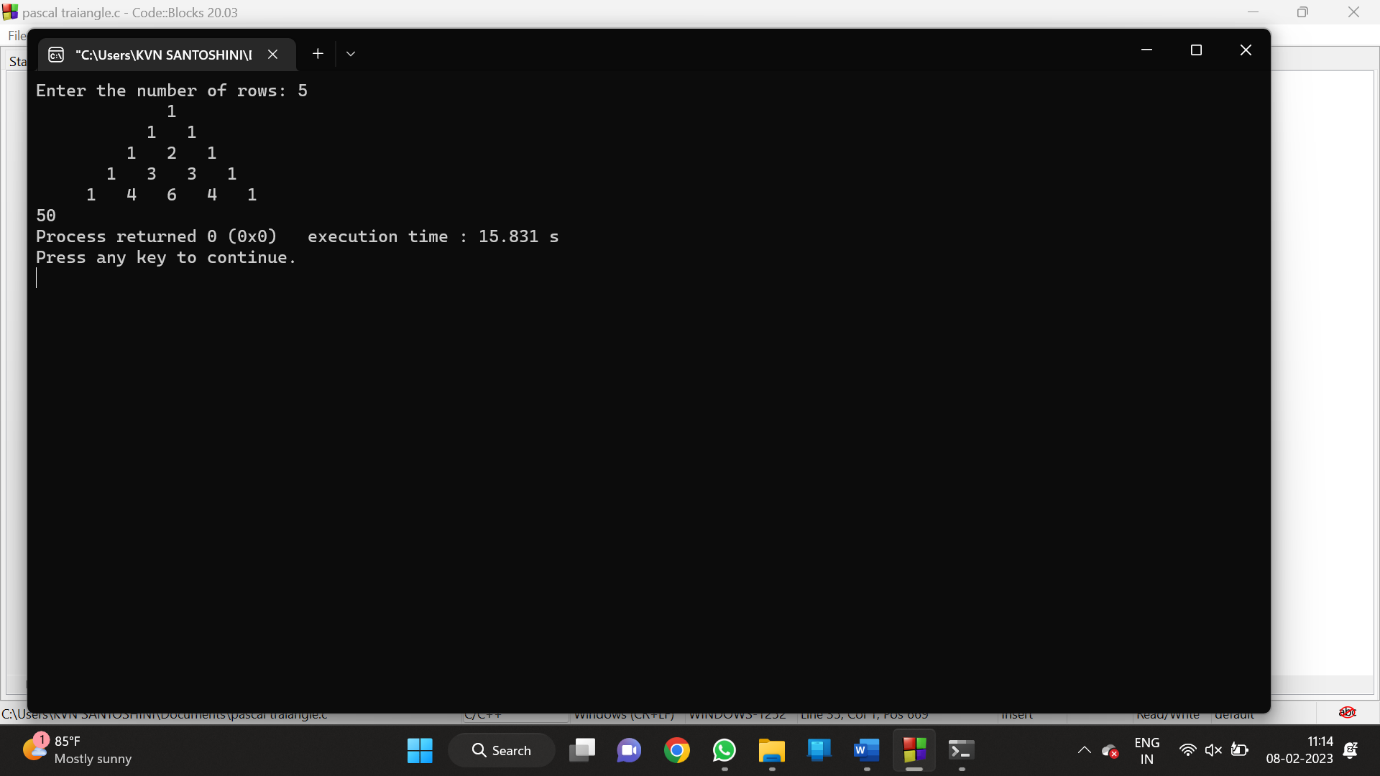
}

printf("%d",count);

return 0;

}

OUTPUT:



7)STREEN MATRIX MULTIPLICATION-TIME COMPLEXITY

PROGRAM:

#include<stdio.h>

int main()

{

int a[2][2], b[2][2], c[2][2], i, j;

int m1, m2, m3, m4 , m5, m6, m7;

int count=0;

printf("Enter the 4 elements of first matrix: ");

count++;

for(i = 0;i < 2; i++)

{

count++;

for(j = 0;j < 2; j++){

count++;

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

}

}

count++;

count++;

printf("Enter the 4 elements of second matrix: ");

for(i = 0; i < 2; i++){

count++;

for(j = 0;j < 2; j++){

count++;

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

}

}

count++;

printf("\nThe first matrix is\n");

for(i = 0; i < 2; i++){

count++;

printf("\n");

for(j = 0; j < 2; j++){

count++;

printf("%d\t", a[i][j]);

}

}

count++;

count++;

printf("\nThe second matrix is\n");

for(i = 0;i < 2; i++){

count++;

printf("\n");

for(j = 0;j < 2; j++){

count++;

printf("%d\t", b[i][j]);

}

}

count++;

count++;

m1= (a[0][0] + a[1][1]) \* (b[0][0] + b[1][1]);

count++;

m2= (a[1][0] + a[1][1]) \* b[0][0];

count++;

m3= a[0][0] \* (b[0][1] - b[1][1]);

count++;

m4= a[1][1] \* (b[1][0] - b[0][0]);

count++;

m5= (a[0][0] + a[0][1]) \* b[1][1];

count++;

m6= (a[1][0] - a[0][0]) \* (b[0][0]+b[0][1]);

count++;

m7= (a[0][1] - a[1][1]) \* (b[1][0]+b[1][1]);

count++;

c[0][0] = m1 + m4- m5 + m7;

count++;

c[0][1] = m3 + m5;

count++;

c[1][0] = m2 + m4;

count++;

c[1][1] = m1 - m2 + m3 + m6;

count++;

printf("\nAfter multiplication using Strassen's algorithm \n");

for(i = 0; i < 2 ; i++){

count++;

printf("\n");

for(j = 0;j < 2; j++){

count++;

printf("%d\t", c[i][j]);

}

}

count++;

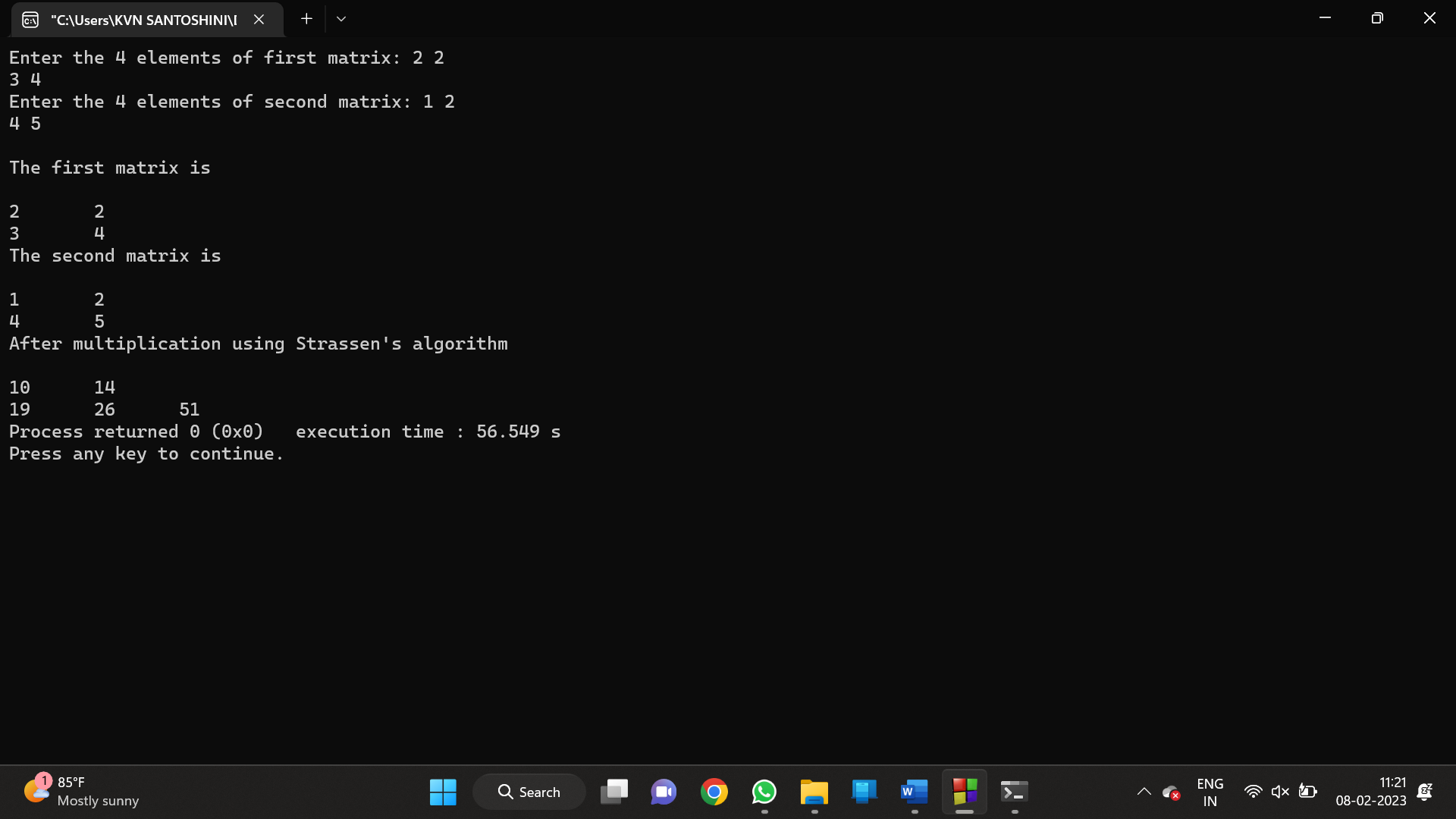
count++;

printf("%d",count);

return 0;

}

OUTPUT:



8) BINARY SEARCH-TIME COMPLEXITY

PROGRAM:

#include <stdio.h>

int main()

{

int i, low, high, mid, n, key, array[100];

int count=0;

printf("Enter number of elements");

scanf("%d",&n);

printf("Enter %d integers", n);

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

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

printf("Enter value to find");

scanf("%d", &key);

low = 0;

count++;

high = n - 1;

count++;

while (low <= high) {

count++;

if(array[mid] < key)

low = mid + 1;

else if (array[mid] == key) {

count++;

printf("%d found at location %d ", key, mid+1);

break;

}

else

high = mid - 1;

mid = (low + high)/2;

count++;

}

if(low > high)

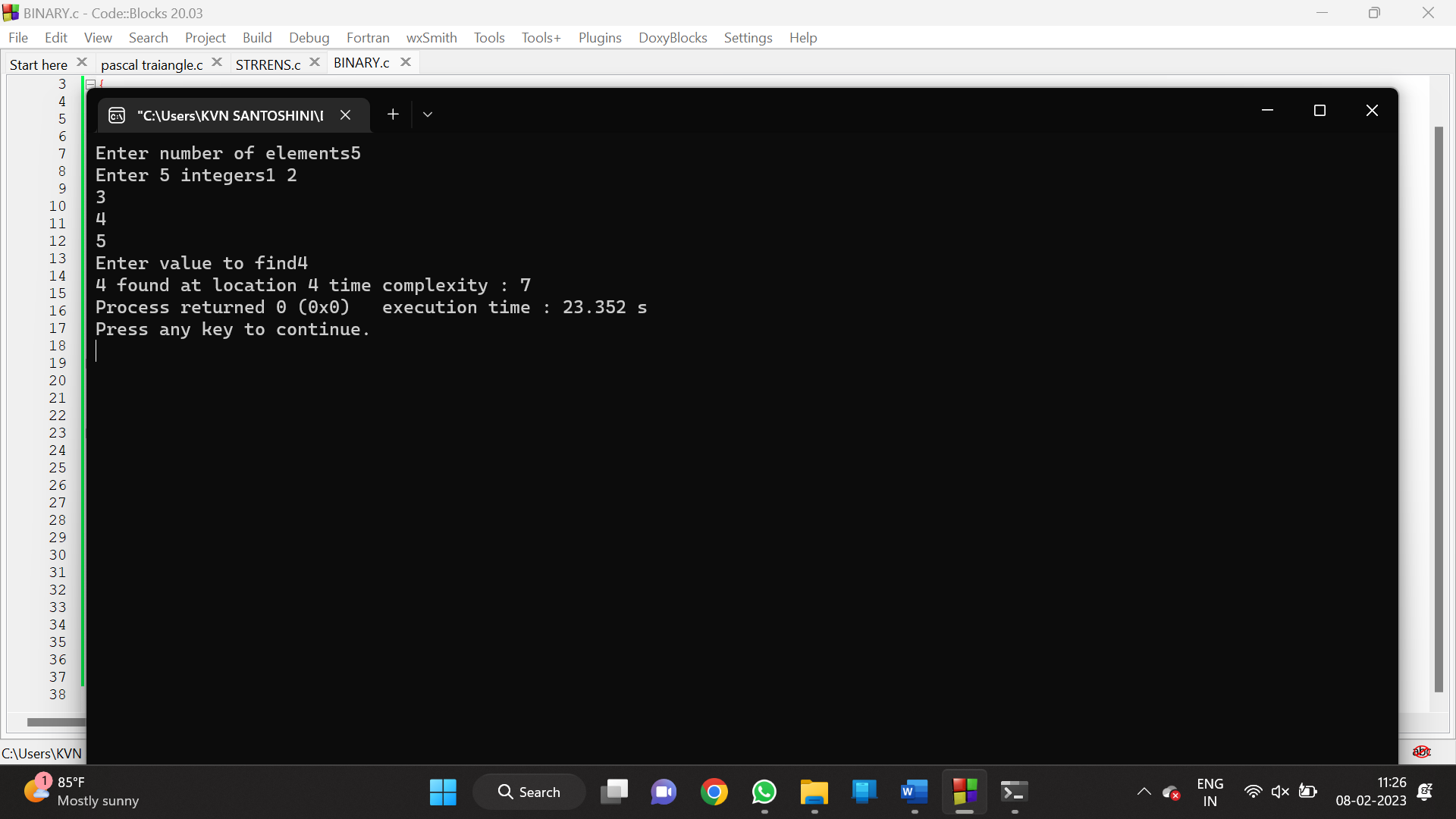
printf("Not found! %d isn't present in the list.n", key);

printf("time complexity : %d",count);

return 0;

}

OUTPUT:



9) LINEAR SEARCH-TIME COMPLEXITY

PROGRAM:

#include<stdio.h>

int main()

{

int array[100], search, c, n;

int count=0;

printf("Enter number of elements in array\n");

scanf("%d", &n);

printf("Enter %d integer(s)\n", n);

for (c = 0; c < n; c++)

{

count++;

scanf("%d", &array[c]);

}

count++;

printf("Enter a number to search\n");

scanf("%d", &search);

for (c = 0; c < n; c++)

{

count++;

if (array[c] == search)

{

printf("%d is present at location %d.\n", search, c+1);

break;

}

}

count++;

if (c == n)

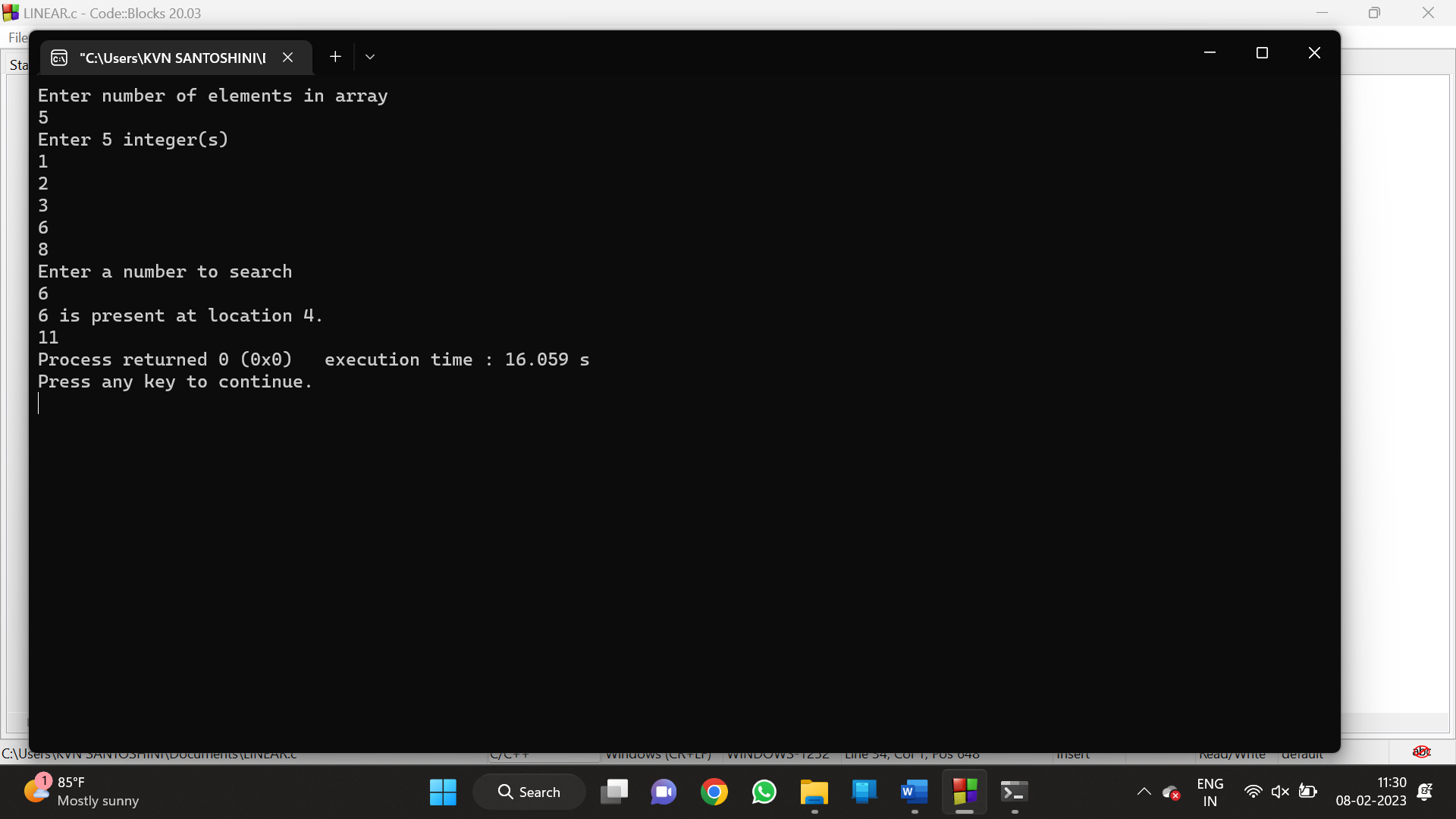
printf("%d isn't present in the array.\n", search);

printf("%d",count);

return 0;

}

OUTPUT:



10) GCD-TIME COMPLEXITY

PROGRAM:

#include <stdio.h>

int main()

{

int n1, n2, i, GCD\_Num;

int count=0;

printf ( " Enter any two numbers: \n ");

scanf ( "%d %d", &n1, &n2);

for( i = 1; i <= n1 && i <= n2; ++i)

{

count++;

if (n1 % i ==0 && n2 % i == 0)

GCD\_Num = i;

count++;

}

count++;

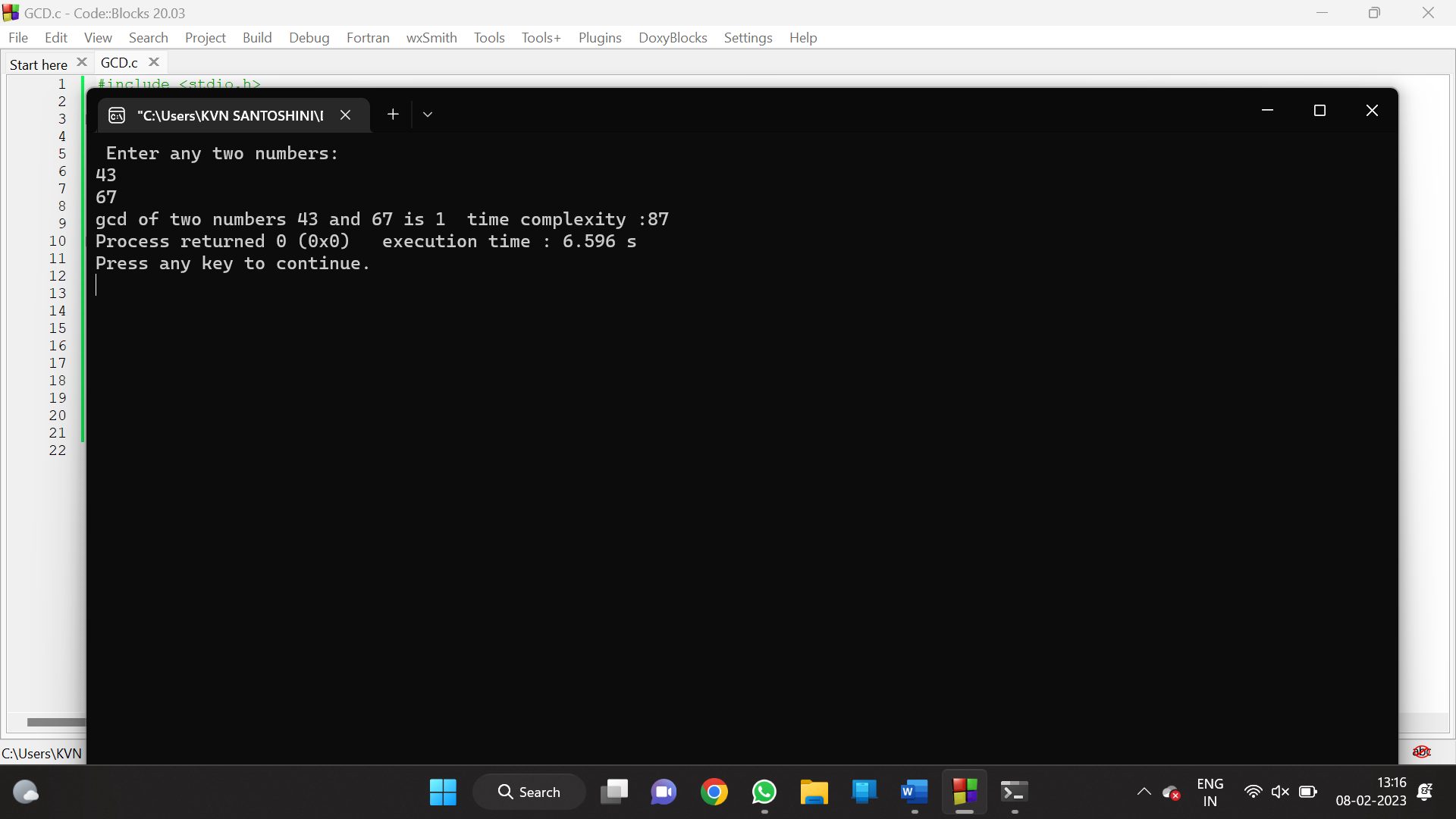
printf ("gcd of two numbers %d and %d is %d ", n1, n2, GCD\_Num);

printf("time complexity :%d ",count);

return 0;

}

OUTPUT:



11) LARGEST NUMBER IN ARRAY-TIME COMPLEXITY

PROGRAM:

#include <stdio.h>

int main() {

int n;

int count=0;

double arr[100];

printf("Enter the number of elements (1 to 100): ");

scanf("%d", &n);

count++;

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

count++;

printf("Enter number%d: ", i + 1);

scanf("%lf", &arr[i]);

}

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

count++;

if (arr[0] < arr[i]) {

arr[0] = arr[i];

}

count++;

}

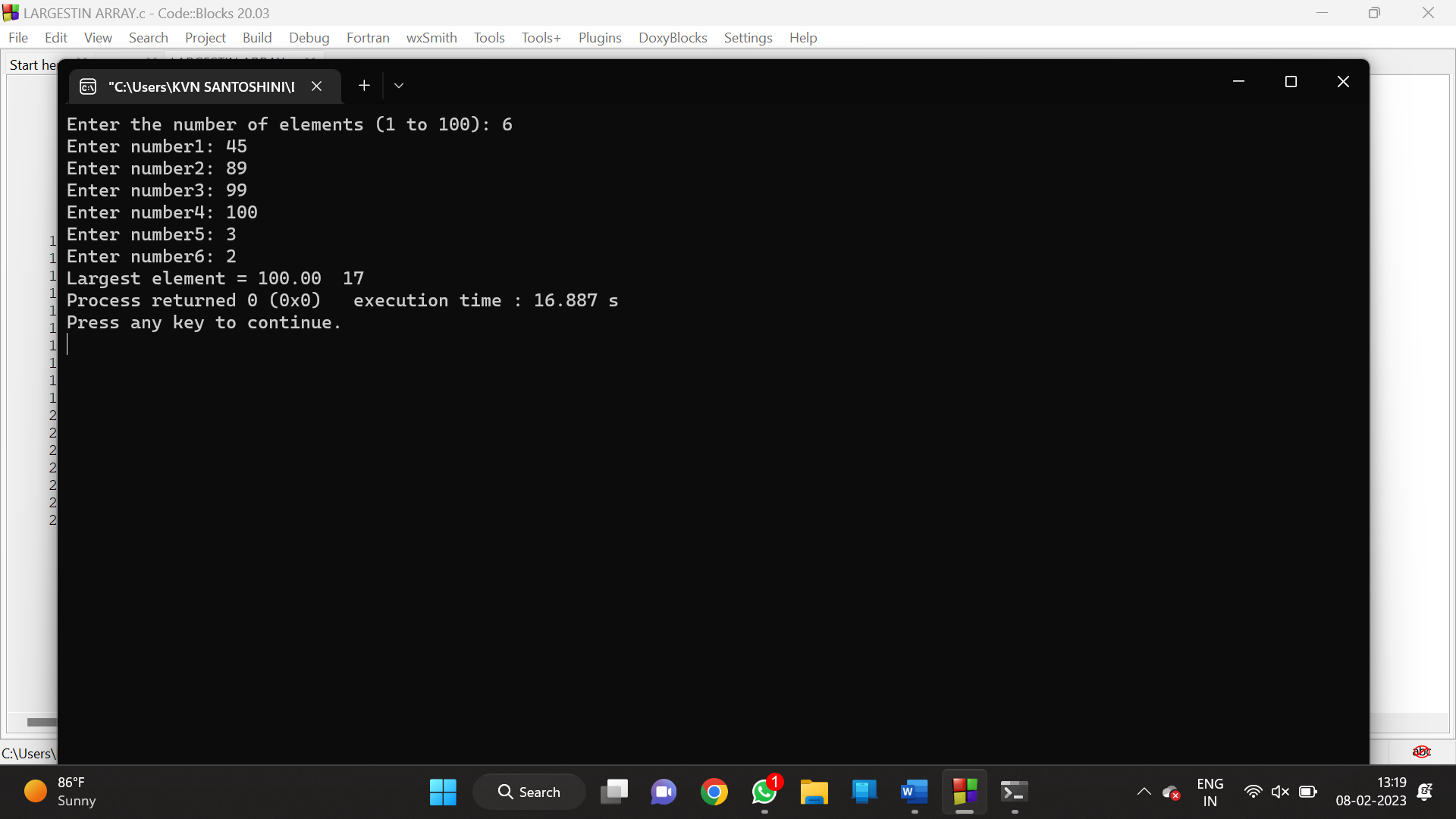
printf("Largest element = %.2lf ", arr[0]);

printf("%d",count);

return 0;

}

OUTPUT:



12) BUBBLE SORT-TIME COMPLEXITY

PROGRAM:

#include<stdio.h>

int main()

{

int ele,count=0;

printf("Enter tot element: ");

scanf("%d",&ele);

int arr[ele];

printf("Enter the elements: ");

for (int i = 0; i < ele; i++){

count++;

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

}count++;

for (int i = 0; i < ele; i++)

{

count++;

for (int j =i+1; j < ele; j++)

{

count++;

if (arr[i]>arr[j])

{

count++;

int temp=arr[i];

count++;

arr[i]=arr[j];

count++;

arr[j]=temp;

count++;

}

}count++;

}count++;

printf("sorted array: ");

for (int i = 0; i < ele; i++)

{count++;

count++;

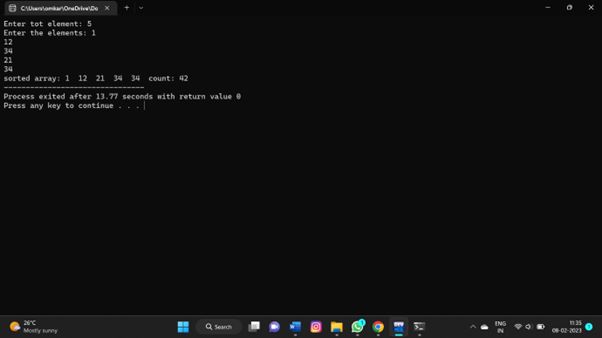
printf("%d ",arr[i]);

}count++;

printf("count: %d",count);

}

OUTPUT:



13) N PERFECT NUMBERS- TIME COMPLEXITY

PROGRAM:

#include <stdio.h>

#include<math.h>

int count=0;

int isPerfect(long long int n) {

long long int dsum = 0;

long long int i;

count++;

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

count++;

if (n % i == 0) {

count++;

if (i == n / i) {

dsum += i;

}

else {

dsum += i;

dsum += n / i;

count++;

}

count++;

}

count++;

}

count++;

dsum = dsum - n;

count++;

if (dsum == n) return 1;

else return 0;

}

int isPrime(long long int n) {

if (n == 1)

return 0;

for (int i = 2; i <= sqrt(n); ++i) {

count++;

if (n % i == 0)

return 0;

}

return 1;

count++;

}

int main() {

long long int n, i, temp;

printf("Enter n: ");

scanf("%d", &n);

count++;

i = 1;

while (n > 0) {

count++;

if (isPrime(i) == 1) {

temp = pow(2, i - 1) \* (pow(2, i) - 1);

count++;

if (isPerfect(temp) == 1) {

printf("%d ", temp);

n = n - 1;

count++;

}

}

i = i + 1;

count++;

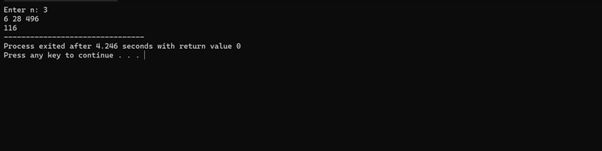
}

printf("\n");

printf("%d",count);

}

OUTPUT:



14) PALINDROME

PROGRAM:

#include <stdio.h>

#include <string.h>

int main() {

char str[100];

int i, length, flag = 0;

printf("Enter a string: ");

scanf("%s", str);

length = strlen(str);

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

if(str[i] != str[length-i-1]){

flag = 1;

break;

}

}

if (flag)

{

printf("%s is not a palindrome and reverse\n", str);

}

else

{

printf("%s is a palindrome\n", str);

}

return 0;

}

OUTPUT:



15) FACTORIAL-TIME COMPLEXITY

PROGRAM:

#include <stdio.h>

int main() {

int n, i;

int count=0;

unsigned long long fact = 1;

printf("Enter an integer: ");

scanf("%d", &n);

count++;

if (n < 0)

printf("Error! Factorial of a negative number doesn't exist.");

else {

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

fact \*= i;

count++;

printf("Factorial of %d = %llu ", n, fact);

printf(" time compexity : %d ",count);

}

return 0;

}

OUTPUT:



16) REVERSE OF NUMBER:

PROGRAM:

#include<stdio.h>

int main()

{

int n,rem,rev;

printf("enter the number:");

scanf("%d",&n);

while(n!=0)

{

rem=n%10;

rev=rev\*10+rem;

n=n/10;

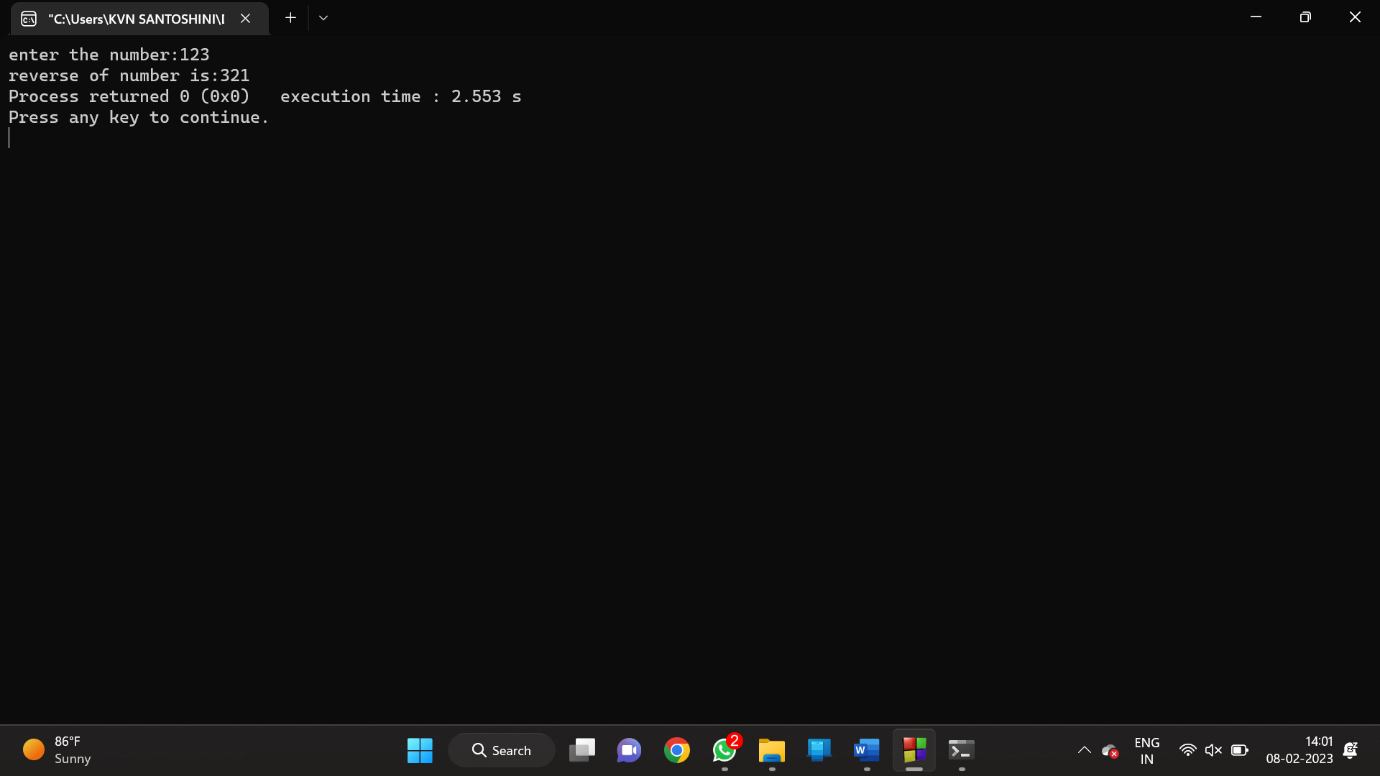
}

printf("reverse of number is:%d",rev);

return 0;

}

OUTPUT:



17) REVERSE OF STRING -TIME COMPLEXITY

PROGRAM:

#include<stdio.h>

int main(){

char val[25];

printf("enter the string: ");

scanf("%s",&val);

int count=0,c=0;

while (val[count]!='\0'){

count++;

c++;

}c++;

for(int i=count-1;i>=0;i--){

c++;

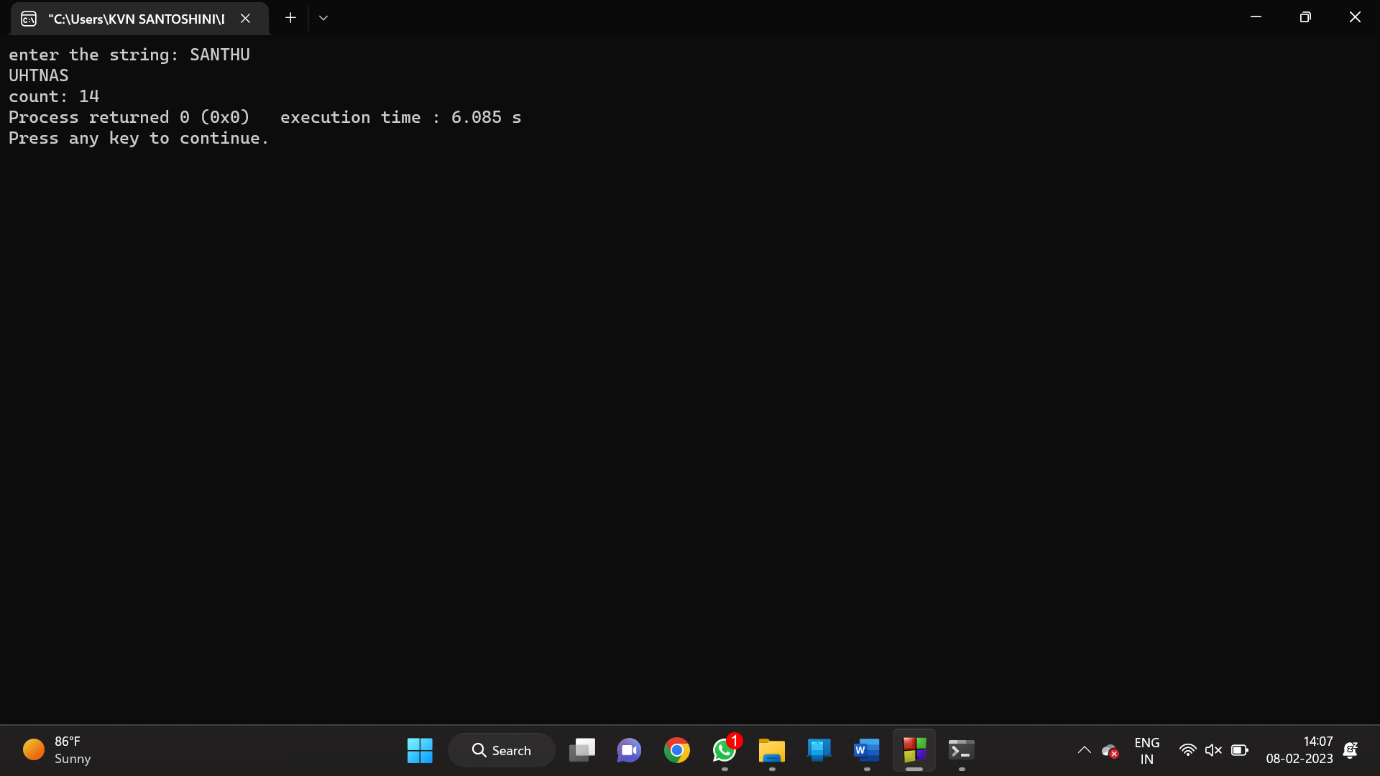
printf("%c",val[i]);

}c++;

printf("\ncount: %d",c);

}

OUTPUT:



18) PRINT SUBSTRING-TIME COMPLEXITY

PROGRAM:

#include<stdio.h>

int main()

{

char str[80], search[10];

int count1 = 0, count2 = 0, i, j, flag;

int count=0;

printf("Enter a string:");

gets(str);

printf("Enter search substring:");

gets(search);

while (str[count1] != '\0')

count1++;

while (search[count2] != '\0')

count2++;

for (i = 0; i <= count1 - count2; i++)

{

count++;

for (j = i; j < i + count2; j++)

{

count++;

flag = 1;

if (str[j] != search[j - i])

{

count++;

flag = 0;

break;

}

count++;

}

if (flag == 1)

break;

count++;

}

count++;

if (flag == 1)

printf("found");

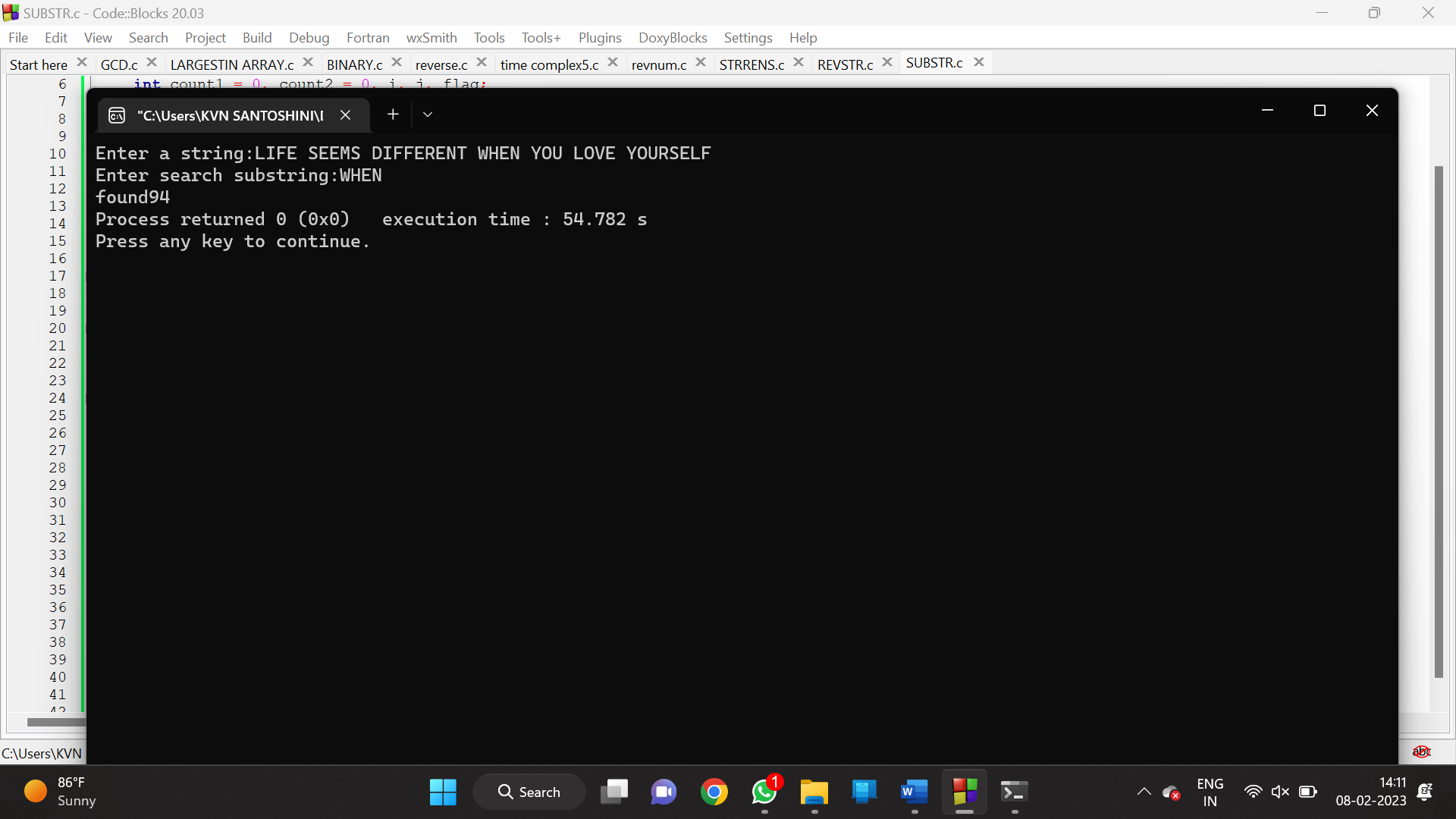
else

printf("not found");

printf("%d",count);

}

OUTPUT:



19) ARMSTRONG NUMBER:

PROGRAM:

#include<stdio.h>

#include<math.h>

int main()

{

int n,r,sum=0,temp,a,c;

printf("enter the number=");

scanf("%d",&n);

temp=n;

a=n;

while(a>0)

{

a=a/10;

c++;

}

while(n>0)

{

r=n%10;

sum+=pow(r,c);

n=n/10;

}

if(temp==sum)

printf("armstrong number ");

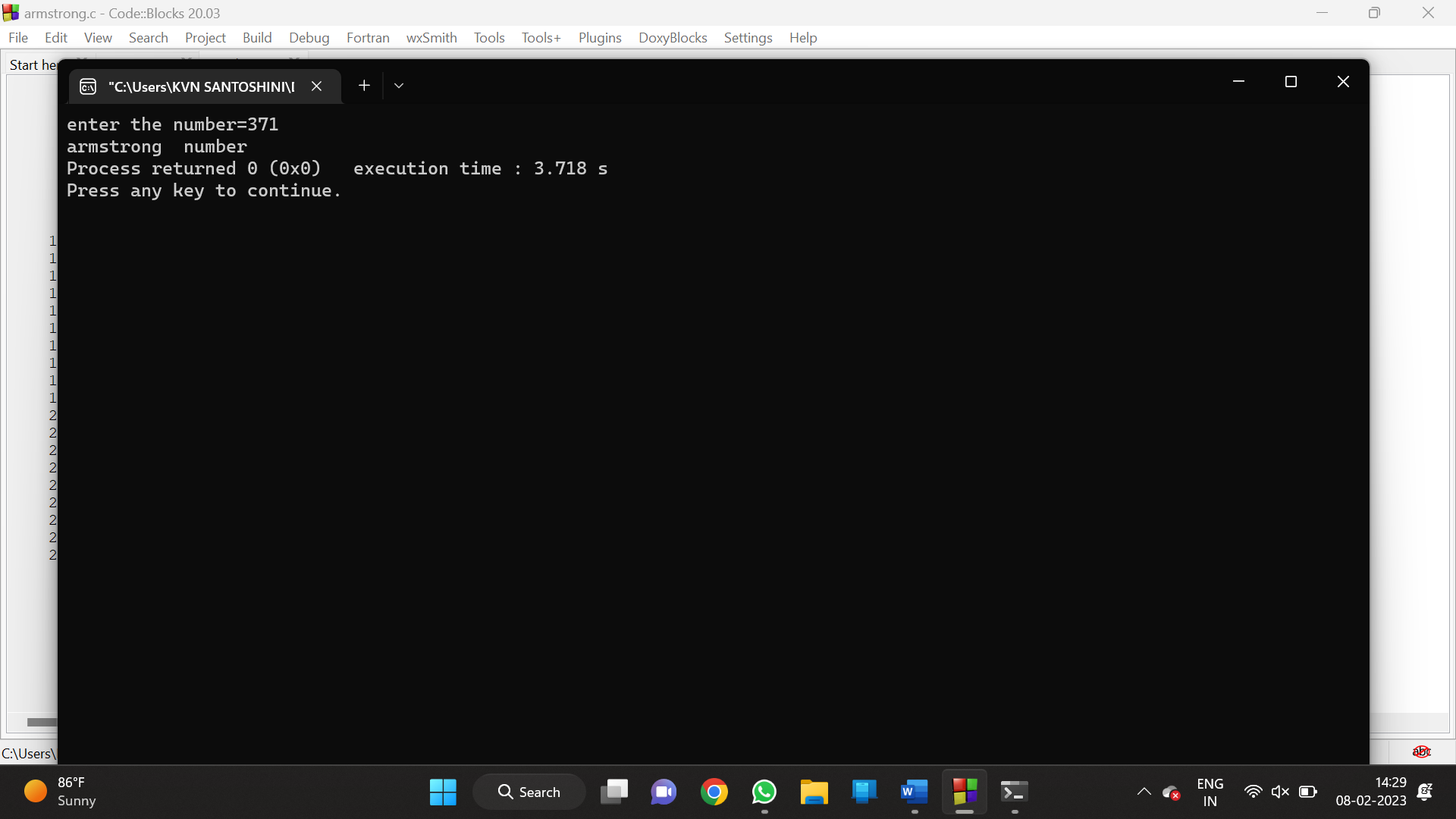
else

printf("not armstrong number");

return 0;

}

OUTPUT:



20) REVERSE PRIME NUMBERS-TIME COMPLEXITY

PROGRAM:

#include<stdio.h>

int main()

{

int c=0;

int n,n1,f,i,j,k,r,p[100],f1;

int sum=0,b=0,rev=0;

c++; c++; c++;

printf("Enter number:");

scanf("%d",&n);

for(j=3;j<=n;j++)

{

c++;

f=0; c++;

for(i=2;i<j;i++)

{

c++;

c++;

if(j%i==0)

{

f=f+1; c++;

}

}

c++;

c++;

if(f==0)

{

n1=j; c++;

rev=0; c++;

while (n1!=0)

{

c++;

r=n1%10; c++;

rev=(rev\*10)+r; c++;

n1=n1/10; c++;

}

c++;

f1=0; c++;

for(k=2;k<rev;k++)

{

c++;

c++;

if(rev%k==0)

{

f1++; c++;

}

}

c++;

c++;

if(f1==0)

{

printf("%d\n",j);

}

}

}

c++;

printf("Time Complexity : %d",c);

}

OUTPUT:

