**1. pointer and array**

#include<stdio.h>

int main()

{

int arr1[5];

char arr2[5];

double arr3[5];

int \*ptr1;

ptr1=arr1;

char \*ptr2;

ptr2=arr2;

double \*ptr3;

ptr3=arr3;

printf("\nsizeof of arr1 of int is=%d\n",sizeof(arr1));

printf("sizeof of ptr1 of int is=%d\n",sizeof(ptr1));

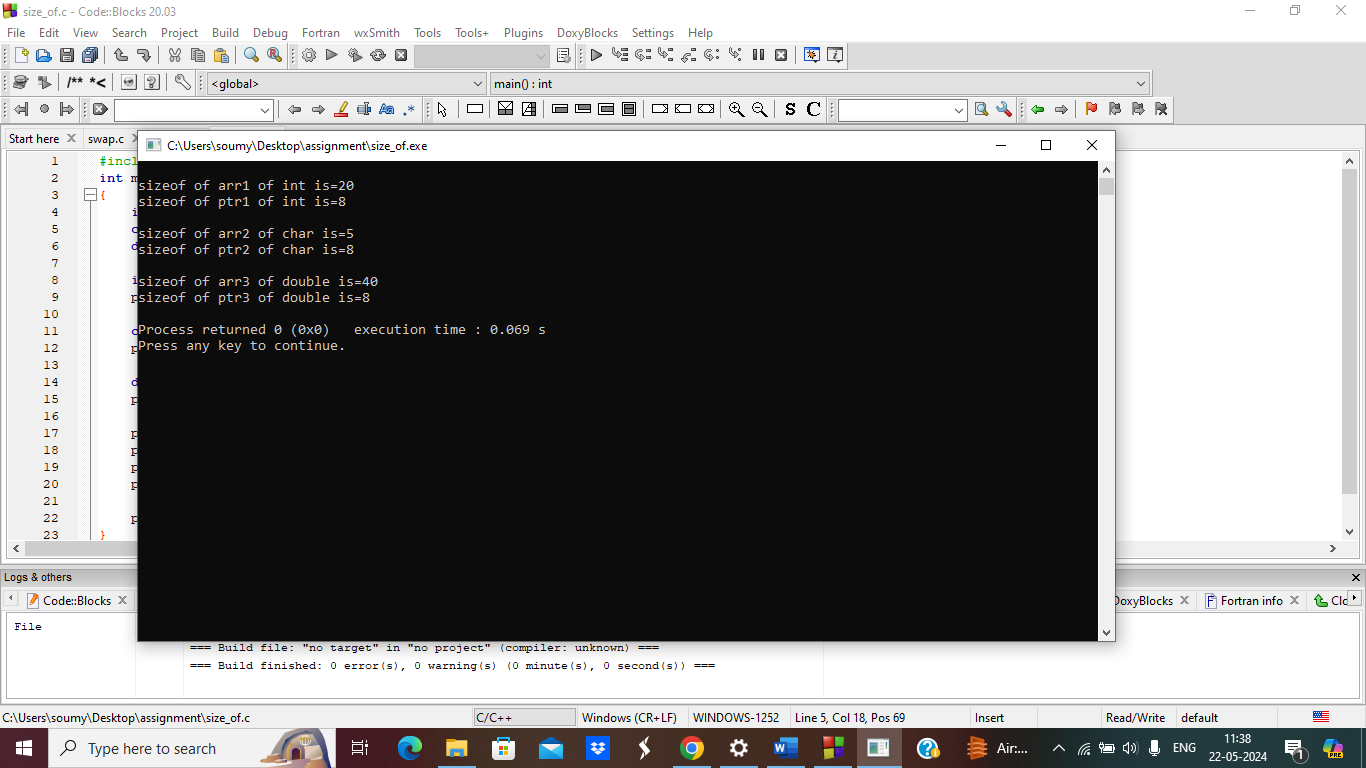
printf("\nsizeof of arr2 of char is=%d\n",sizeof(arr2));

printf("sizeof of ptr2 of char is=%d\n",sizeof(ptr2));

printf("\nsizeof of arr3 of double is=%d\n",sizeof(arr3));

printf("sizeof of ptr3 of double is=%d\n",sizeof(ptr3));

}



**2**.

#include <stdio.h>

int main() {

int array[5];

int \*ptr;

ptr = array;

printf("Size of the array : %zu bytes\n", sizeof(array));

printf("Size of the pointer : %zu bytes\n", sizeof(ptr));

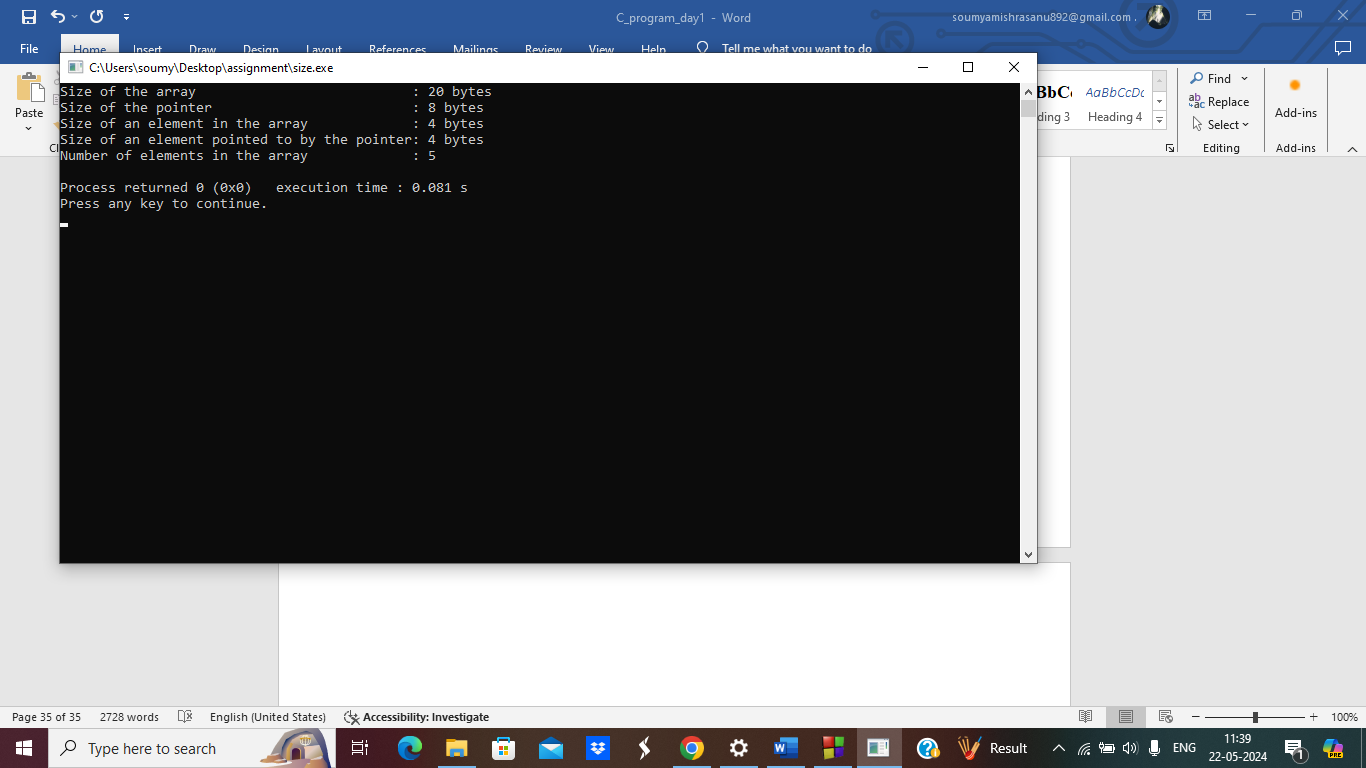
printf("Size of an element in the array : %zu bytes\n", sizeof(array[0]));

printf("Size of an element pointed to by the pointer: %zu bytes\n", sizeof(ptr[0]));

printf("Number of elements in the array : %zu\n", sizeof(array) / sizeof(array[0]));

return 0;

}



**3. Size of a structure**

#include <stdio.h>

struct bag

{

char x;

char y;

int z;

};

int main()

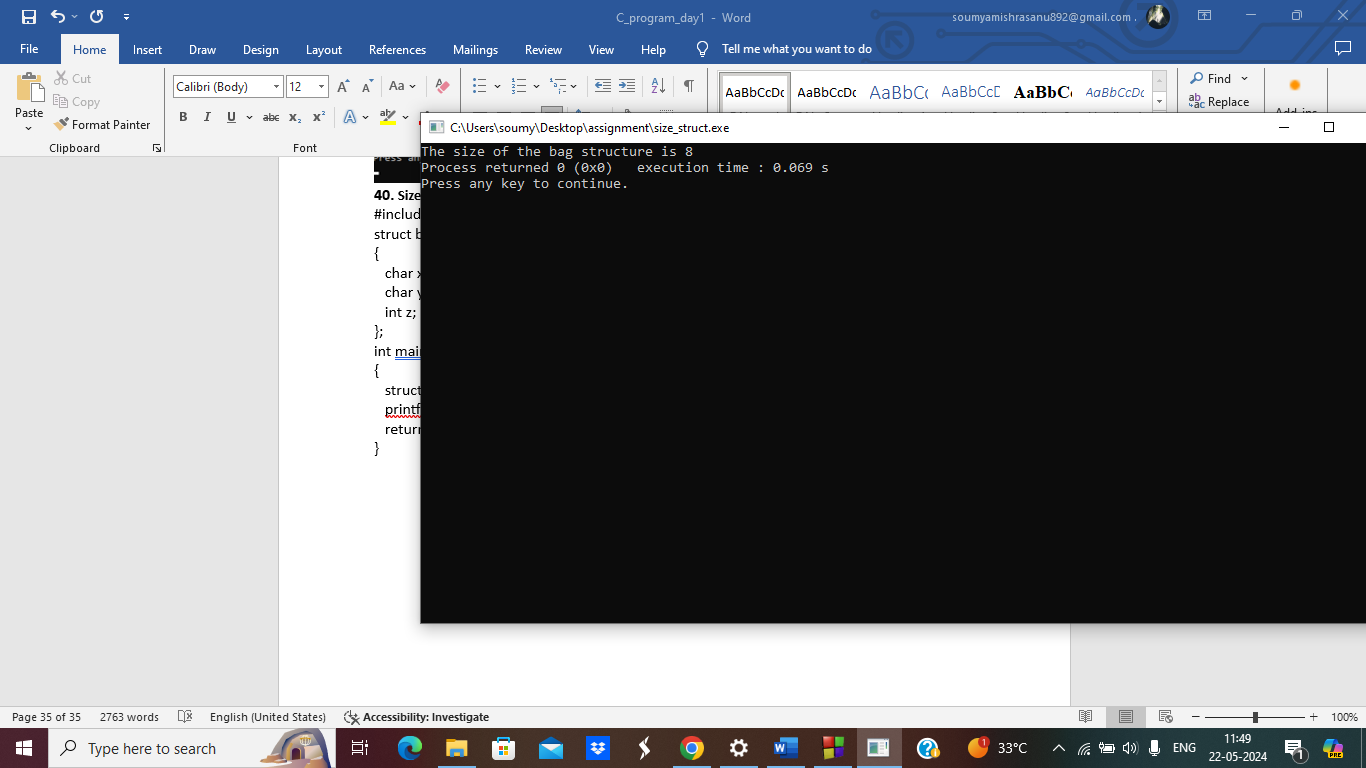
{

struct bag bag1;

printf("The size of the bag structure is %d", sizeof(bag1));

return 0;

}



**4.**

#include <stdio.h>

struct bag

{

char x;

int z;

char y;

};

int main()

{

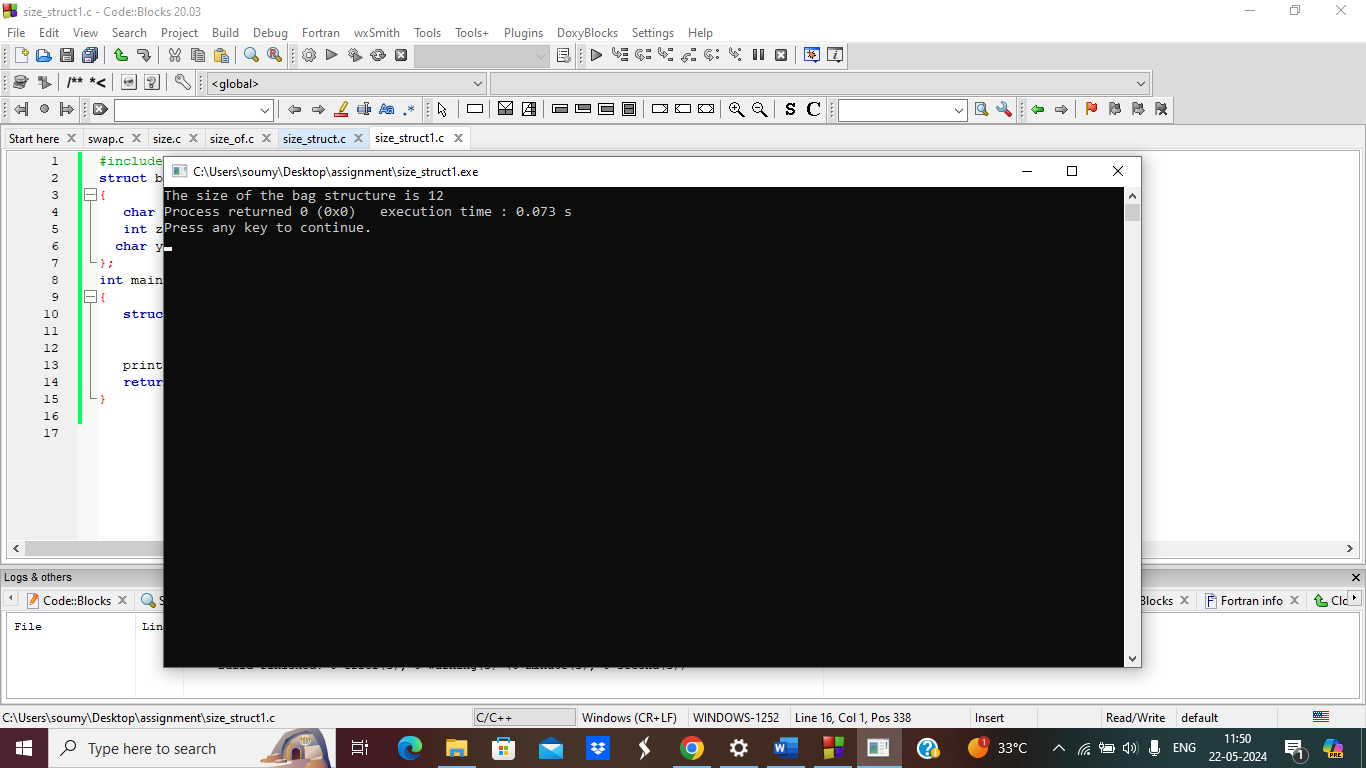
struct bag bag1; // variable is declared of the object types..

// size of the structured bag is displayed.

printf("The size of the bag structure is %d", sizeof(bag1)); // size of bag is printed

return 0;

}



**5.with structure padding**

#include <stdio.h>

struct bag

{

int x;

char y;

double z;

};

int main()

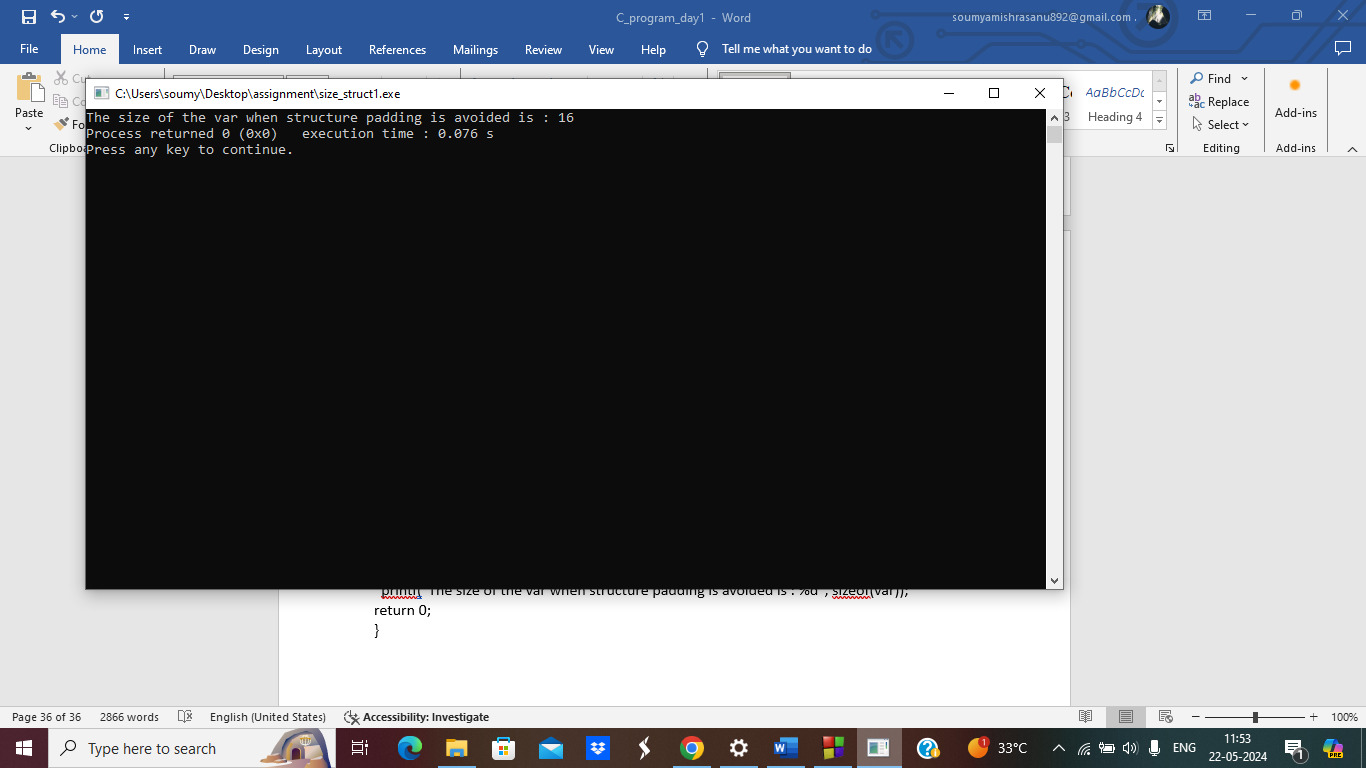
{

struct bag var;

printf("The size of the var when structure padding is avoided is : %d", sizeof(var));

return 0;

}



**6.using #pragma pack(1) Directive: to avoid structure padding**

#include <stdio.h>

struct bag

{

int x;

char y;

double z;

};

int main()

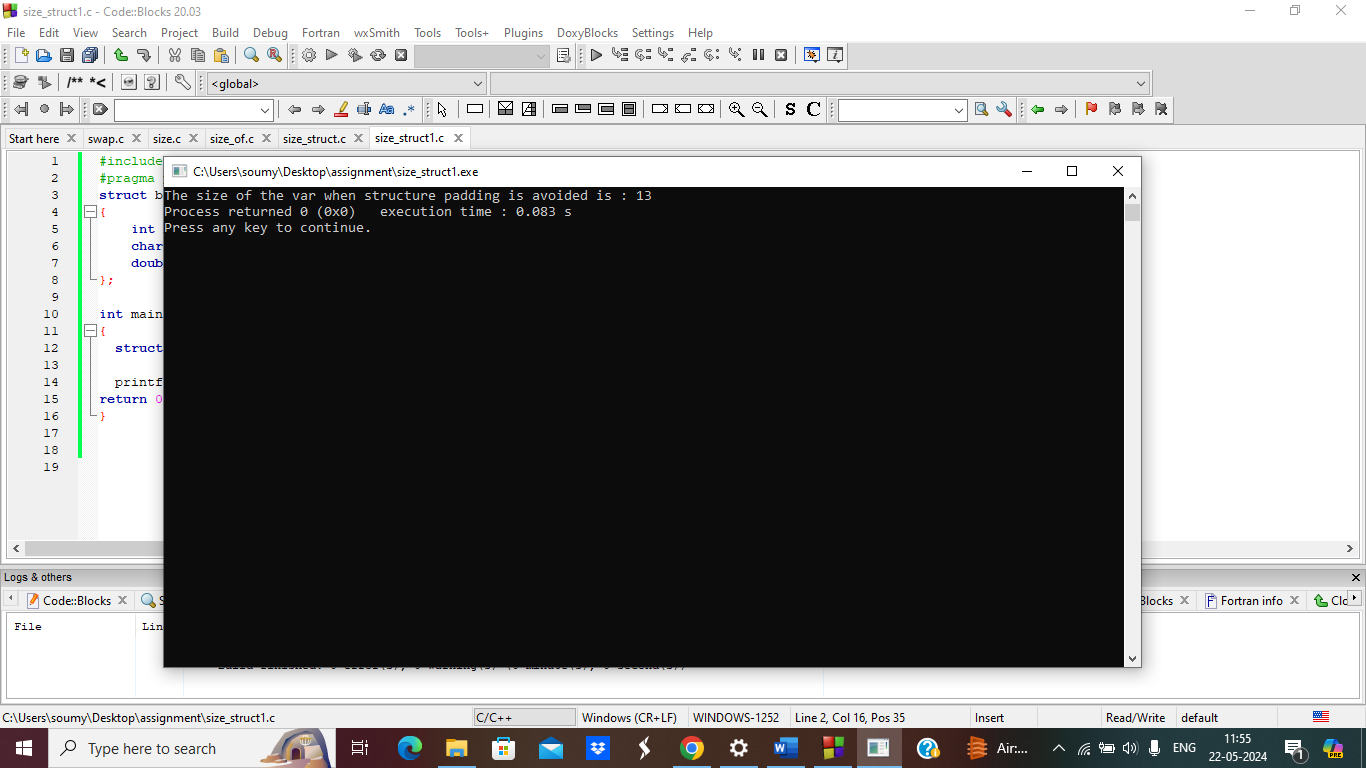
{

struct bag var;

printf("The size of the var when structure padding is avoided is : %d", sizeof(var));

return 0;

}



**7.check prime number**

#include <stdio.h>

int isPrime(int num) {

if (num < 2)

return 0;

for (int i = 2; i \* i <= num; i++) {

if (num % i == 0)

return 0;

}

return 1;

}

int main() {

int num;

printf("Enter a number: ");

scanf("%d", &num);

if (isPrime(num))

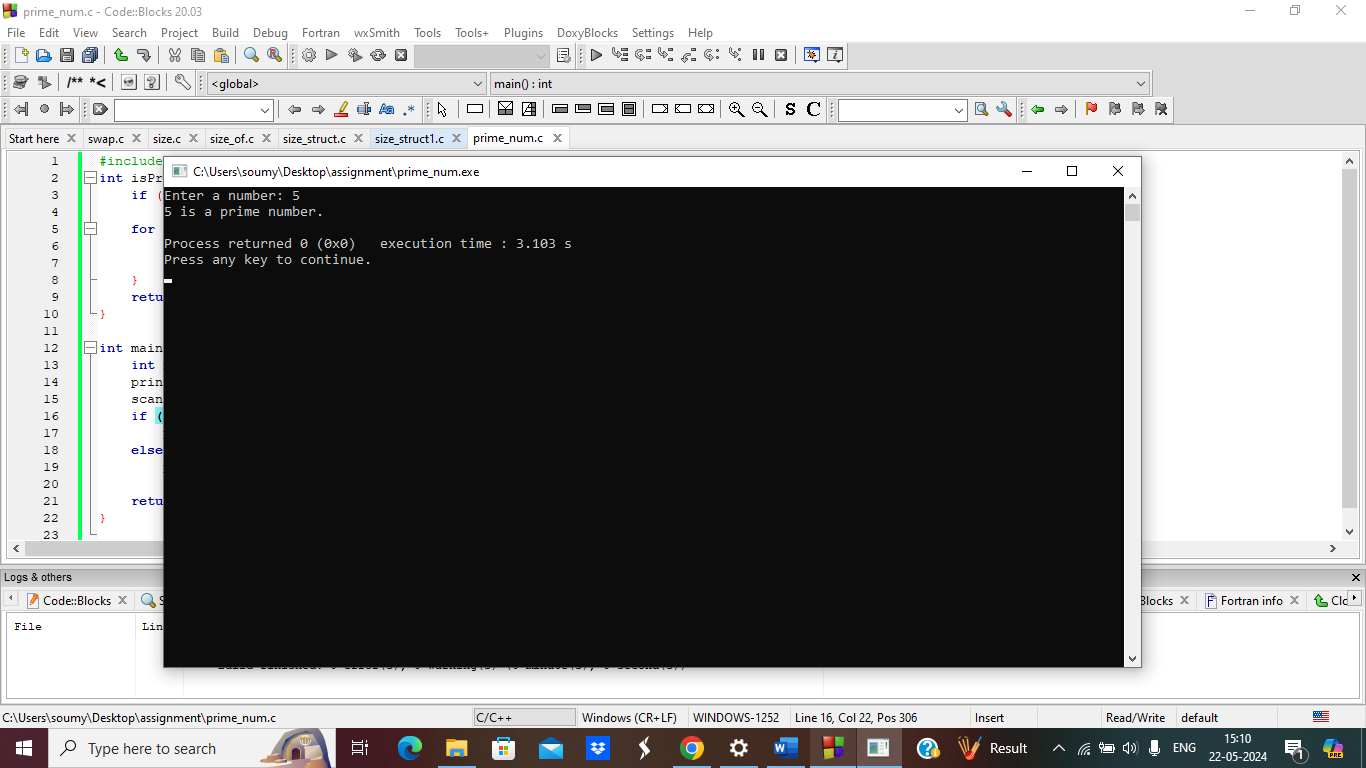
printf("%d is a prime number.\n", num);

else

printf("%d is not a prime number.\n", num);

return 0;

}



**8.find factorial using recursion.**

#include<stdio.h>

int main()

{

int i,fact=1,number;

printf("Enter a number: ");

scanf("%d",&number);

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

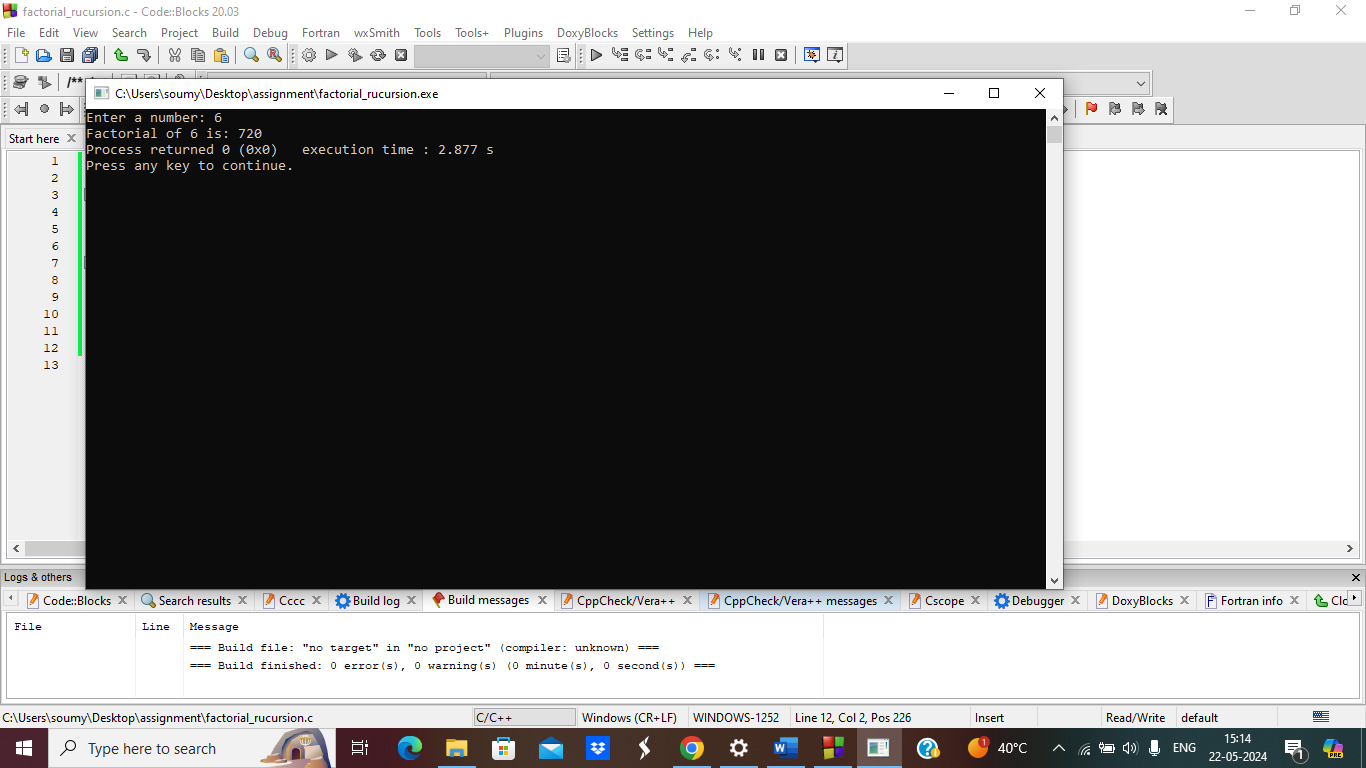
fact=fact\*i;

}

printf("Factorial of %d is: %d",number,fact);

return 0;

}



**9.fibonacci using recursion.**

#include <stdio.h>

int fibonacci(int n) {

if(n == 0)

return 0;

else if(n == 1)

return 1;

else

return (fibonacci(n-1) + fibonacci(n-2));

}

int main() {

int n;

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

scanf("%d", &n);

printf("Fibonacci Series: ");

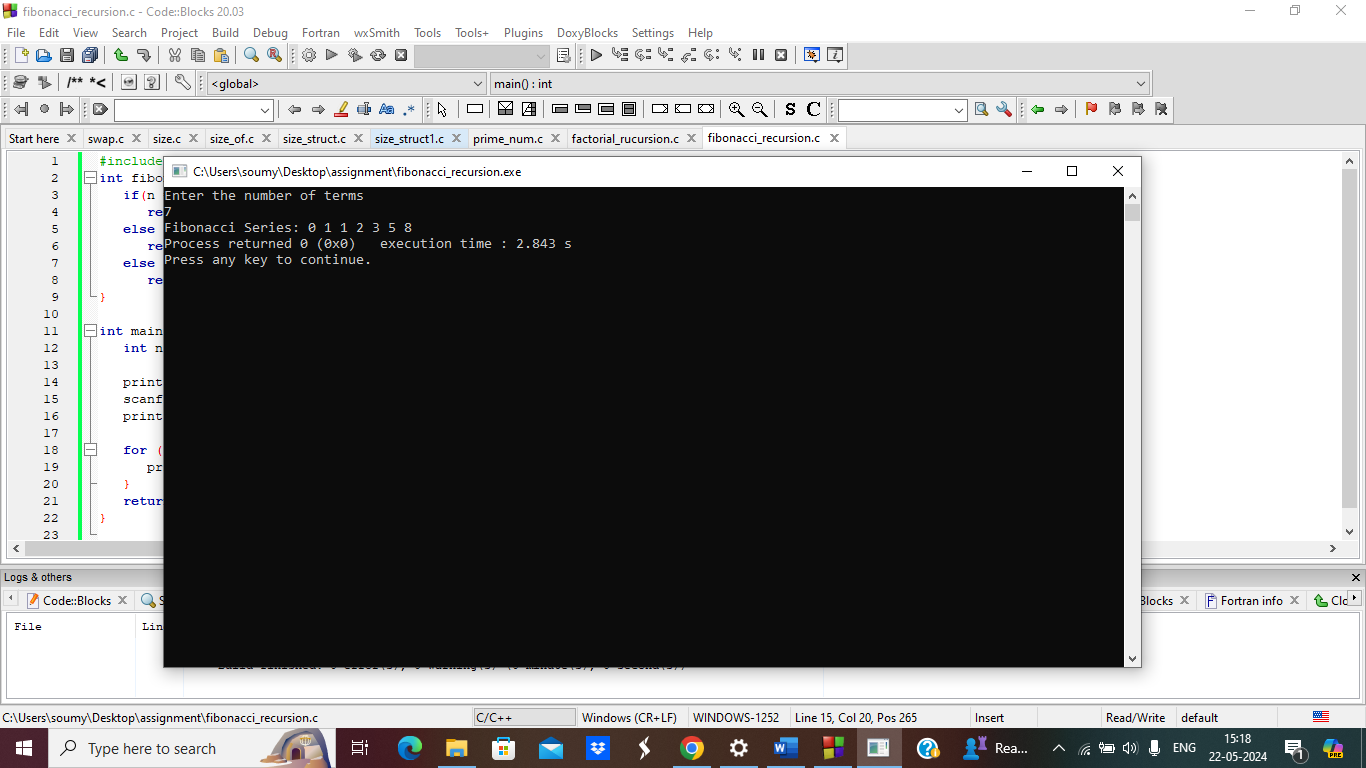
for (int i = 0; i < n; i++) {

printf("%d ", fibonacci(i));

}

return 0;

}



**10.Tower of Hanoi.**

#include <stdio.h>

void hanoi(int n, char from, char to, char via) {

if(n == 1){

printf("Move disk 1 from %c to %c\n", from, to);

}

else{

hanoi(n-1, from, via, to);

printf("Move disk %d from %c to %c\n", n, from, to);

hanoi(n-1, via, to, from);

}

}

int main() {

int n = 3;

char from = 'A';

char to = 'B';

char via = 'C';

hanoi(n, from, via, to);

}

