1.

#include <stdio.h>

int main () {

int value\_array[] = {42, 366, 458, 112};

int a, \*point[4];

printf("TechVidvan Tutorial: Array of pointers!\n\n");

for (a = 0; a < 4; a++) {

point[a] = &value\_array[a]; // assigning address!

}

for (a= 0; a < 4; a++) {

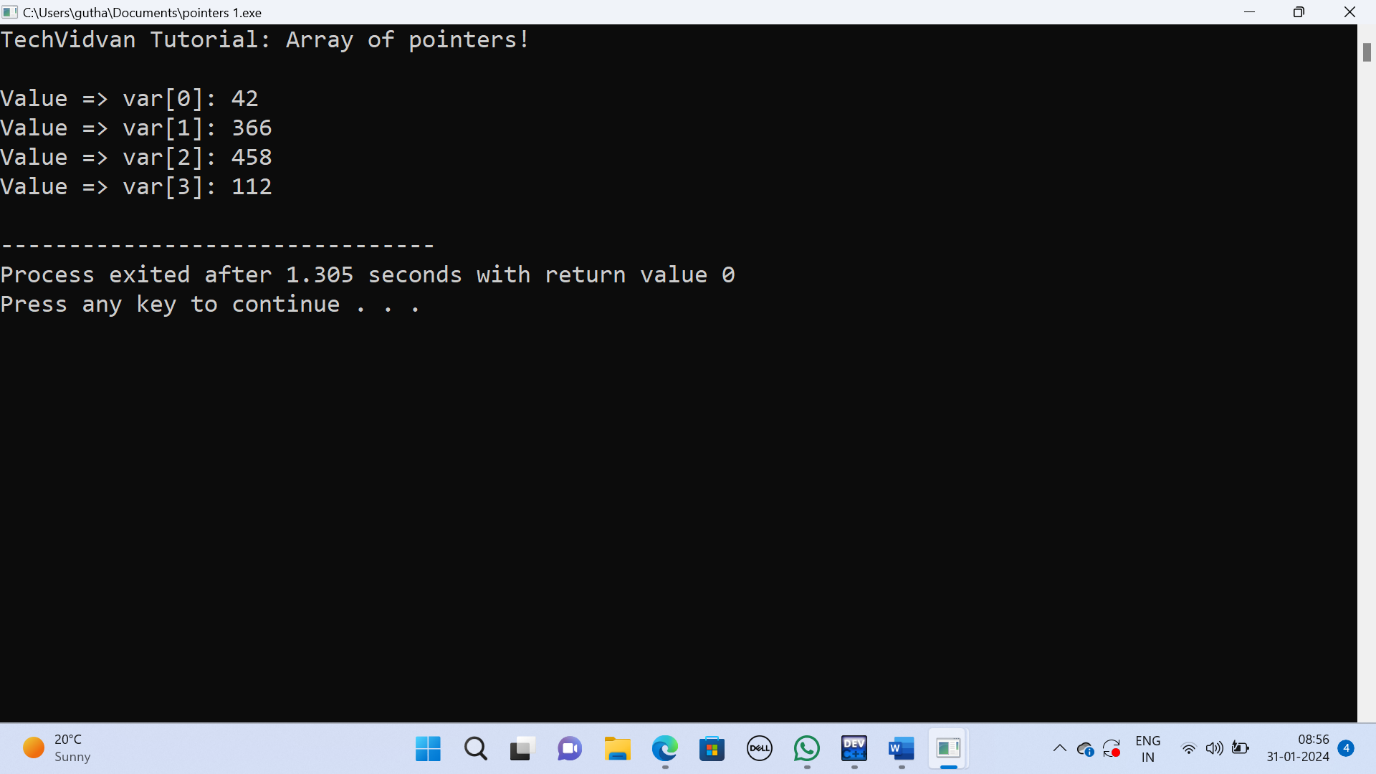
printf("Value => var[%d]: %d\n", a, \*point[a] );

}

return 0;

}

Output:



2.

#include <stdio.h>

int main () {

int val;

int \*pt;

int \*\*point;

val = 55;

pt = &val;

point = &pt;

printf("Value => val: %d\n", val);

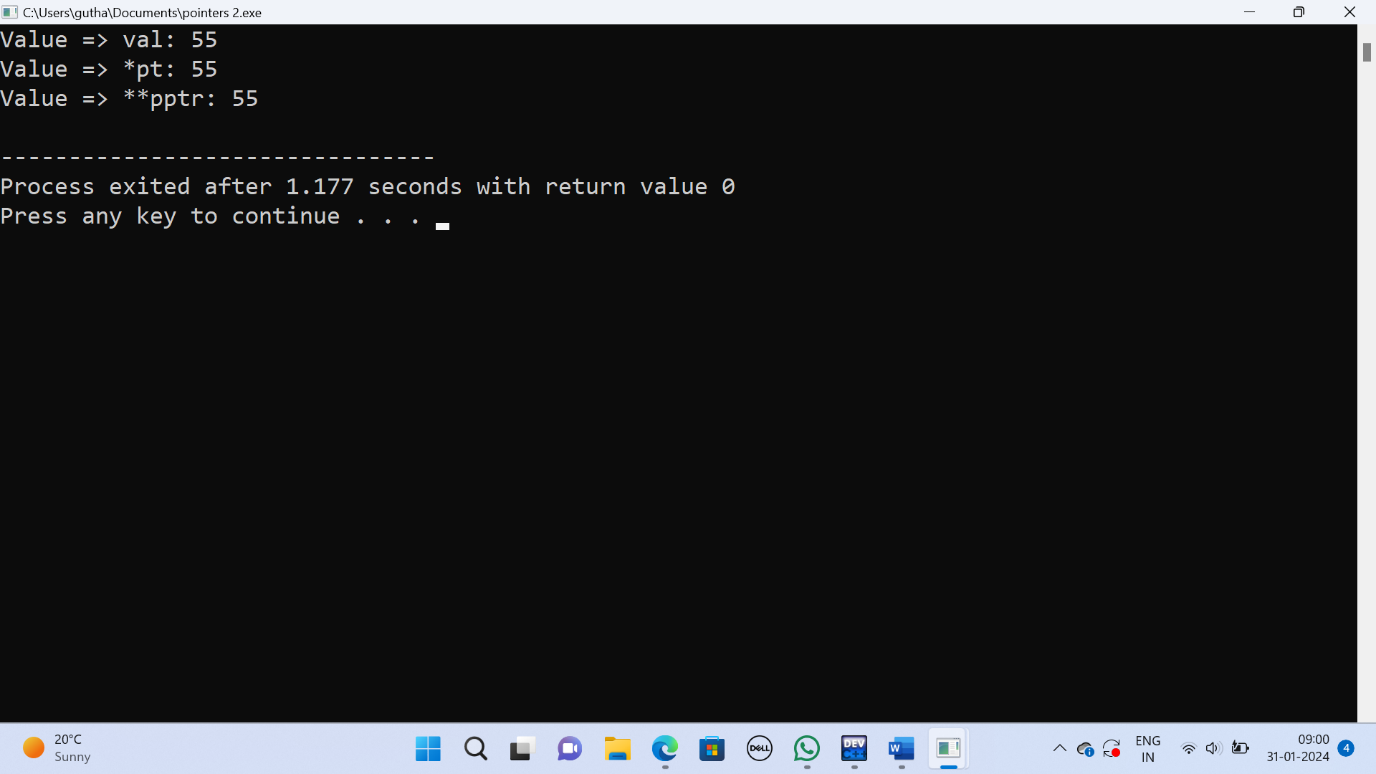
printf("Value => \*pt: %d\n", \*pt );

printf("Value => \*\*pptr: %d\n", \*\*point);

return 0;

}

Output:



3.#include<stdio.h>

#include <conio.h>

int main()

{

int a[6] = {10, 20, 30, 40, 50, 60};

int \*p;

int i;

p = a;

for (i = 0; i < 6; i++)

{

printf("%d ", \*p); // value of elements of array

printf("%u\n", p); // Address of array

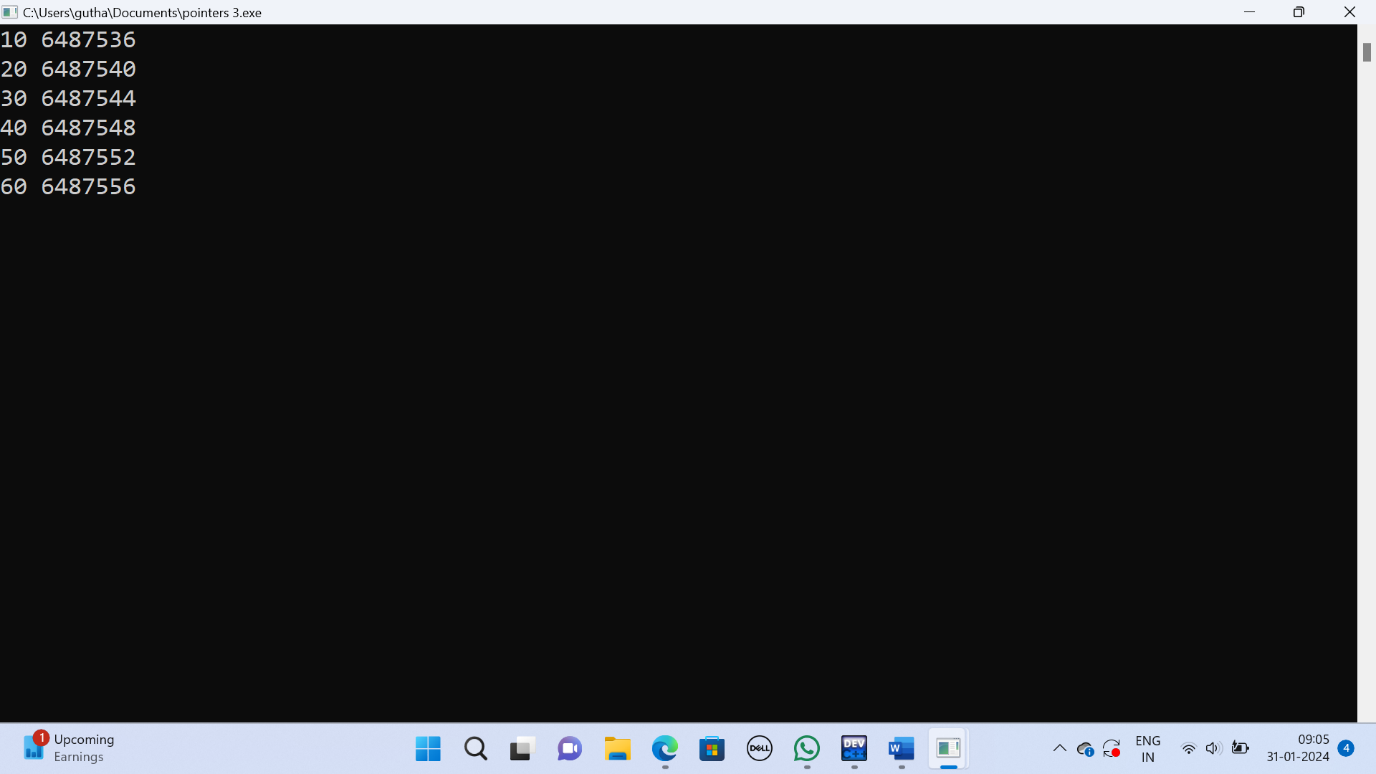
p++; // Move to the next element in the array

}

getch(); // Wait for a key press before exiting

}

Output:



4.

#include<stdio.h>

#include <conio.h>

int main()

{

int a[10];

int i,sum=0;

int \*ptr;

printf("Enter 10 elements:n");

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

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

ptr = a; /\* a=&a[0] \*/

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

{

sum = sum + \*ptr;

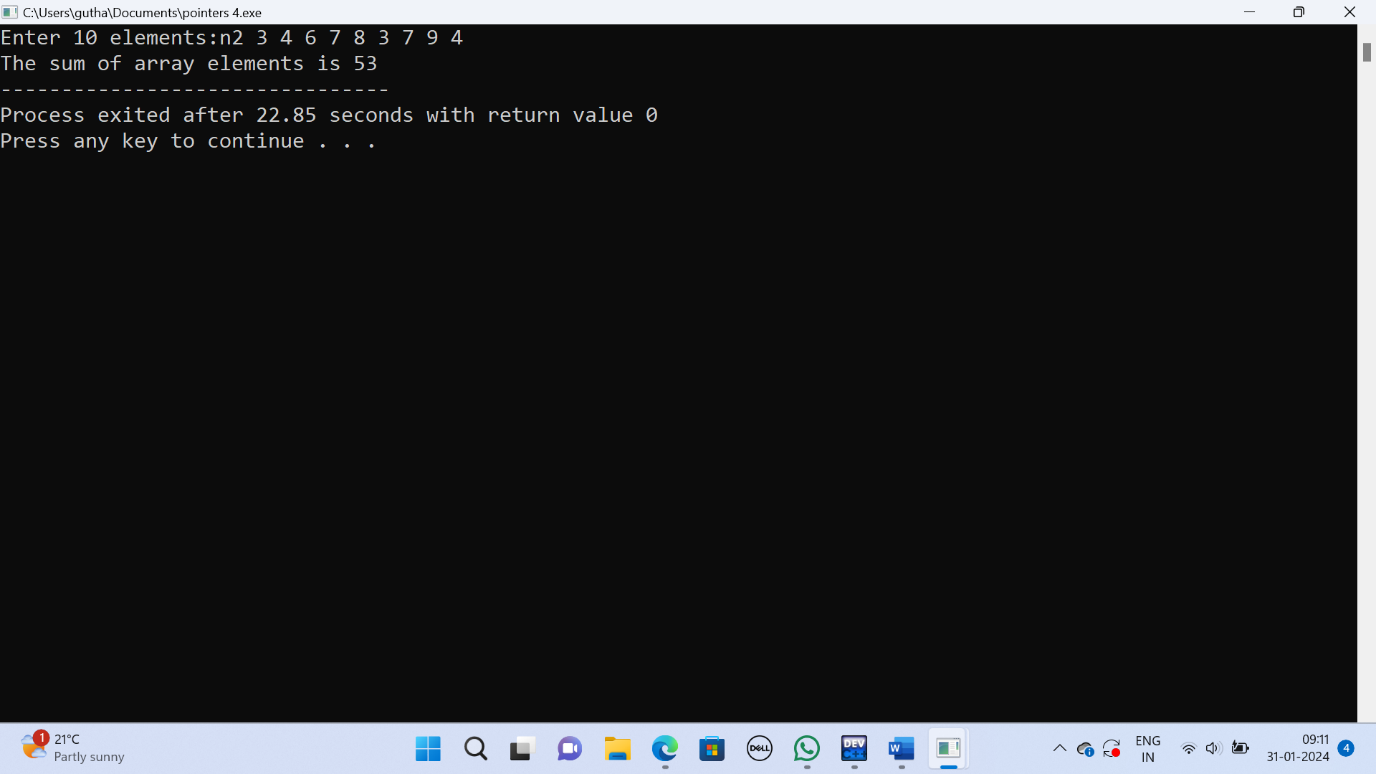
ptr++;

}

printf("The sum of array elements is %d",sum);

}

Output:



5.

#include<stdio.h>

int main() {

int i, j, temp1, temp2;

int arr[8] = {5, 3, 0, 2, 12, 1, 33, 2};

int \*ptr;

// Bubble sort algorithm

for (i = 0; i < 7; i++) {

for (j = 0; j < 7 - i; j++) {

if (\*(arr + j) > \*(arr + j + 1)) {

ptr = arr + j;

temp1 = \*ptr++;

temp2 = \*ptr;

\*ptr-- = temp1;

\*ptr = temp2;

}

}}

for (i = 0; i < 8; i++) {

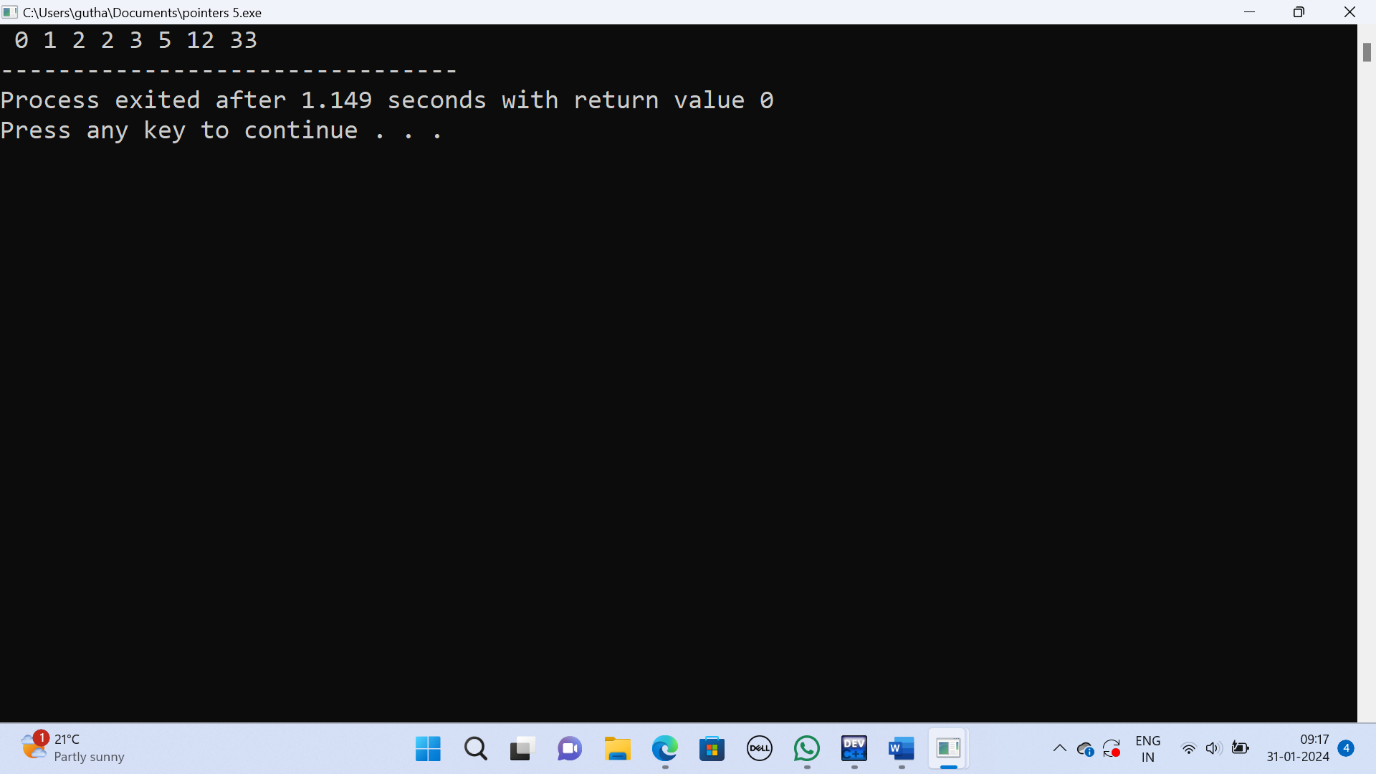
printf(" %d", arr[i]);

}

return 0;

}

Output:



6.

#include <stdio.h>

#include <stdlib.h>

struct node {

int i;

float j;

};

int main() {

struct node \*s[10];

for (int k = 0; k < 10; ++k) {

s[k] = (struct node\*)malloc(sizeof(struct node));

s[k]->i = k;

s[k]->j = 0.5 \* k;

}

for (int k = 0; k < 10; ++k) {

printf("Element %d: i = %d, j = %.2f\n", k, s[k]->i, s[k]->j);

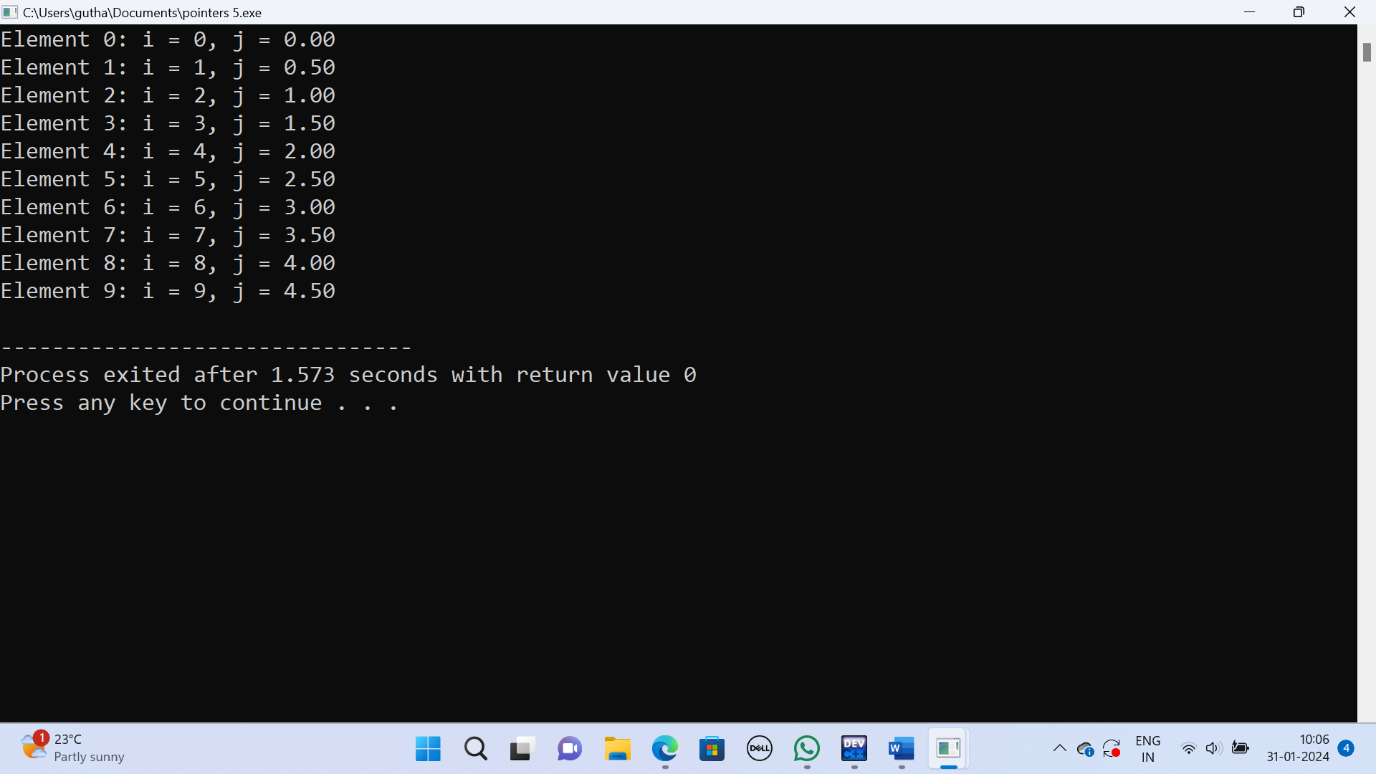
free(s[k]);

}

return 0;

}

Output:



7.

#include <stdio.h>

int main() {

char c[] = "GATE2011";

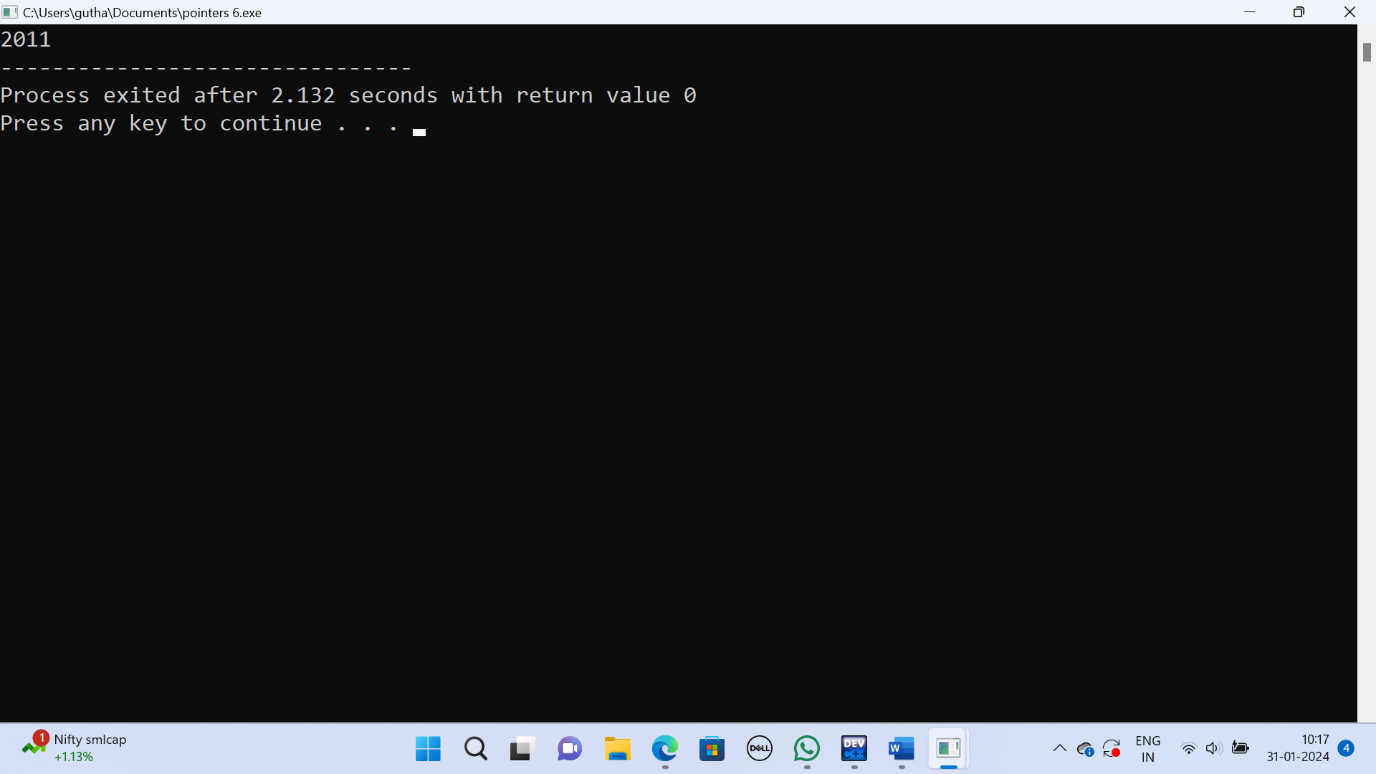
char \*p = c;

printf("%s", p + p[3] - p[1]);

return 0;

}

Output:



8.

#include <stdio.h>

void swap(int \*a, int \*b) {

int temp;

temp = \*a;

\*a = \*b;

\*b = temp;

}

int main() {

int x = 5, y = 10;

printf("Before swapping: x = %d, y = %d\n", x, y);

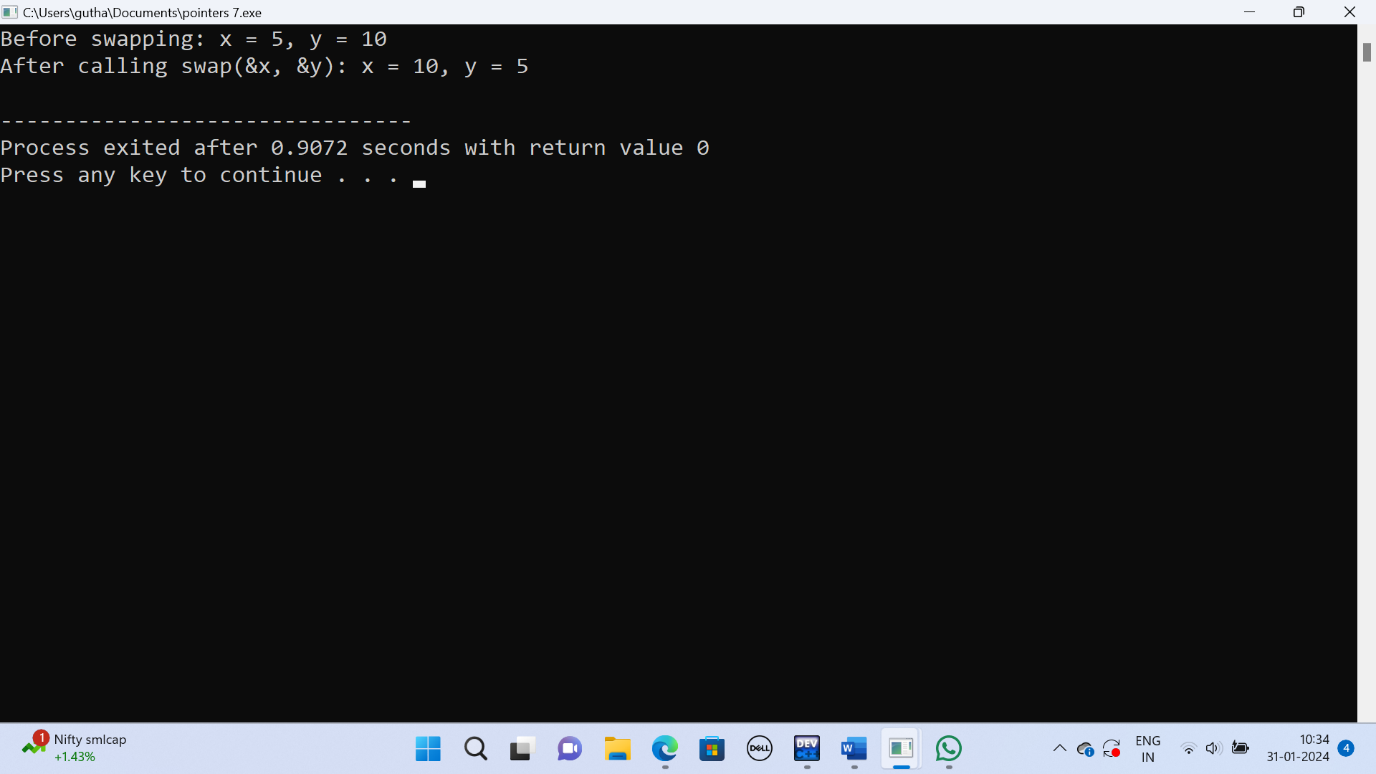
swap(&x, &y);

printf("After calling swap(&x, &y): x = %d, y = %d\n", x, y);

return 0;

}

Output:



9.

#include<stdio.h>

int main() {

int arr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 5};

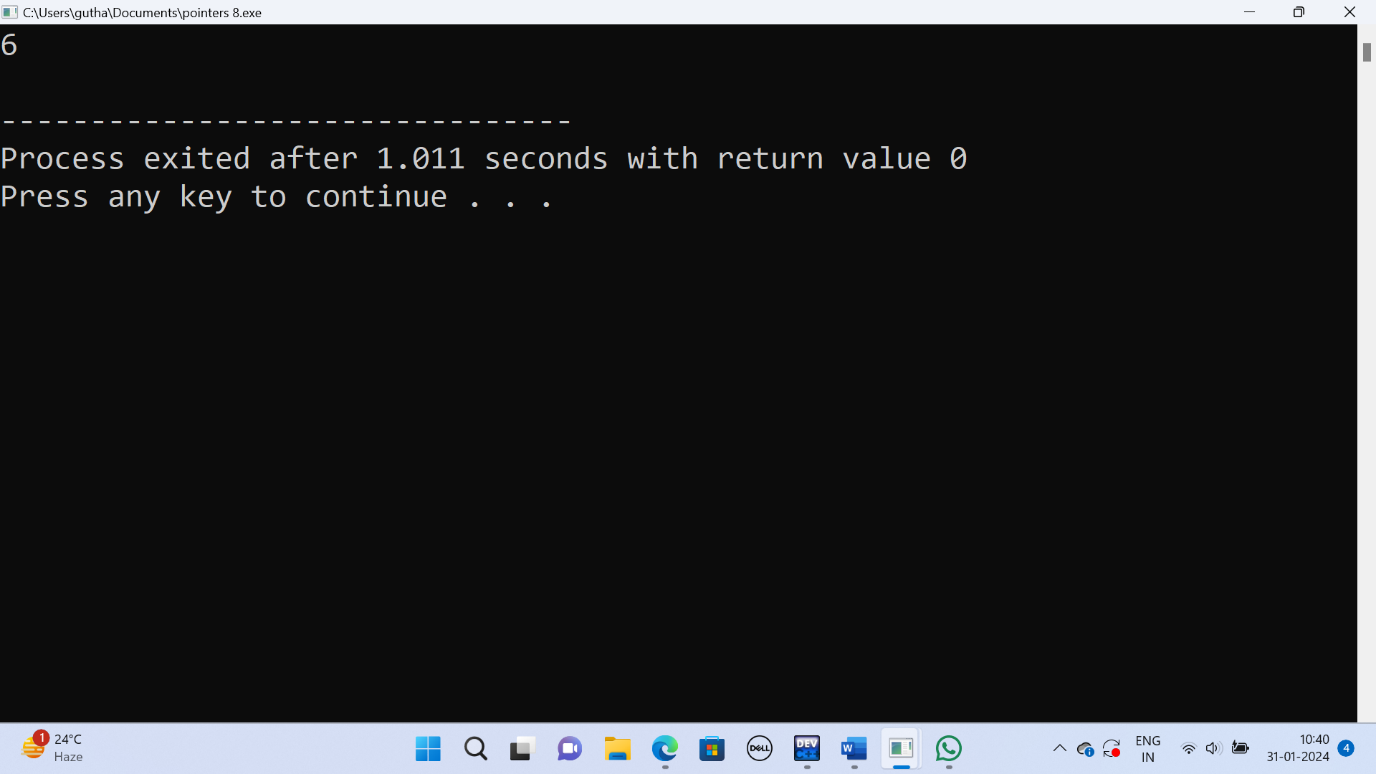
int \*ip = arr + 4;

printf("%d\n", ip[1]);

return 0;

}

Output:



10.

#include <stdio.h>

void mystery(int \*ptra, int \*ptrb) {

int \*temp;

temp = ptrb;

ptrb = ptra;

ptra = temp;

}

int main() {

int a = 2016, b = 0, c = 4, d = 42;

mystery(&a, &b);

if (a < c)

mystery(&c, &a);

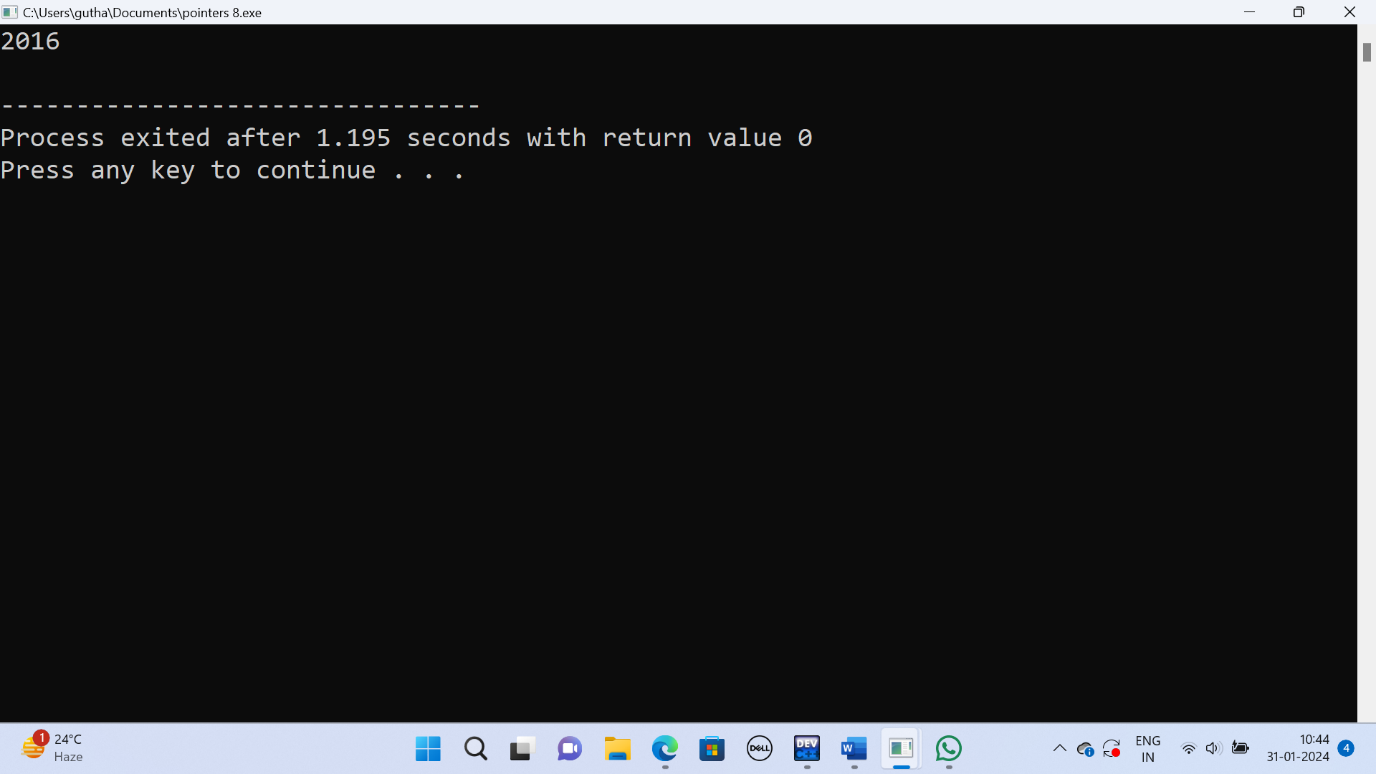
mystery(&a, &d);

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

return 0;

}

Output:



11.

#include <stdio.h>

struct Ournode {

char x, y, z;

};

int main() {

struct Ournode p = {'1', '0', 'a' + 2};

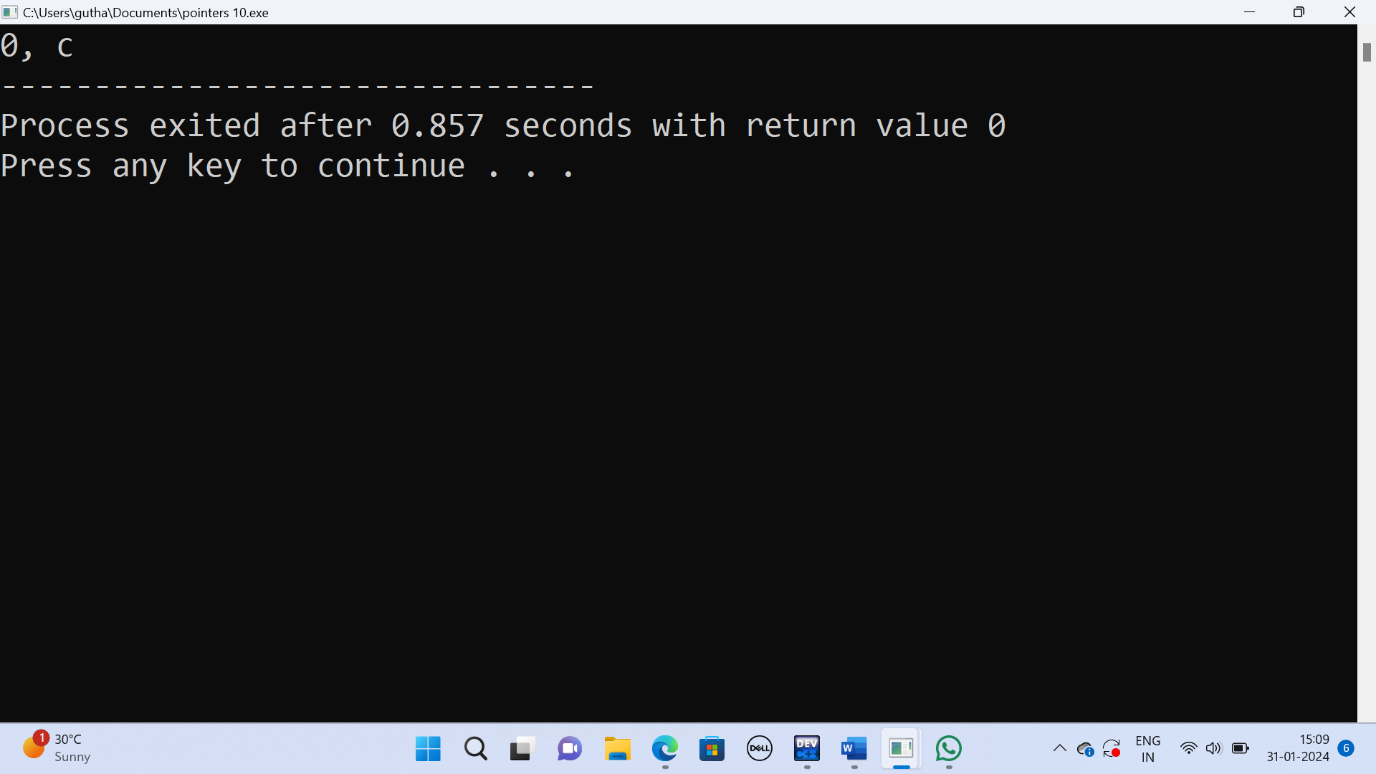
struct Ournode \*q = &p;

printf("%c, %c", \*((char \*)q + 1), \*((char \*)q + 2));

return 0;

}

Output:



12.

#include<stdio.h>

void f(int \*p, int \*q) {

p = q;

\*p = 2;

}

int i = 0, j = 1;

int main() {

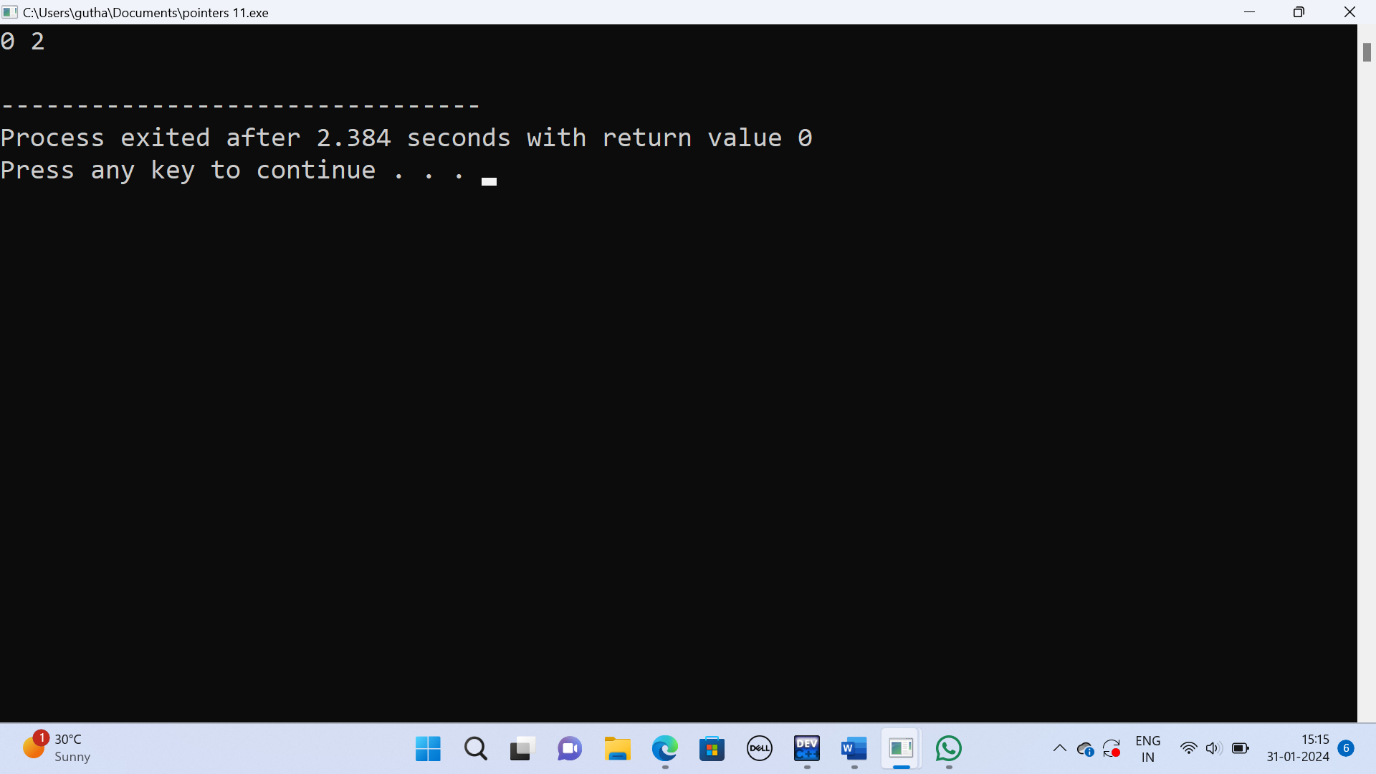
f(&i, &j);

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

return 0;

}

Output:



13.

#include <stdio.h>

void f1(int a, int b) {

int c;

c = a;

a = b;

b = c;

}

void f2(int \*a, int \*b) {

int c;

c = \*a;

\*a = \*b;

\*b = c;

}

int main() {

int a = 4, b = 5, c = 6;

f1(a, b);

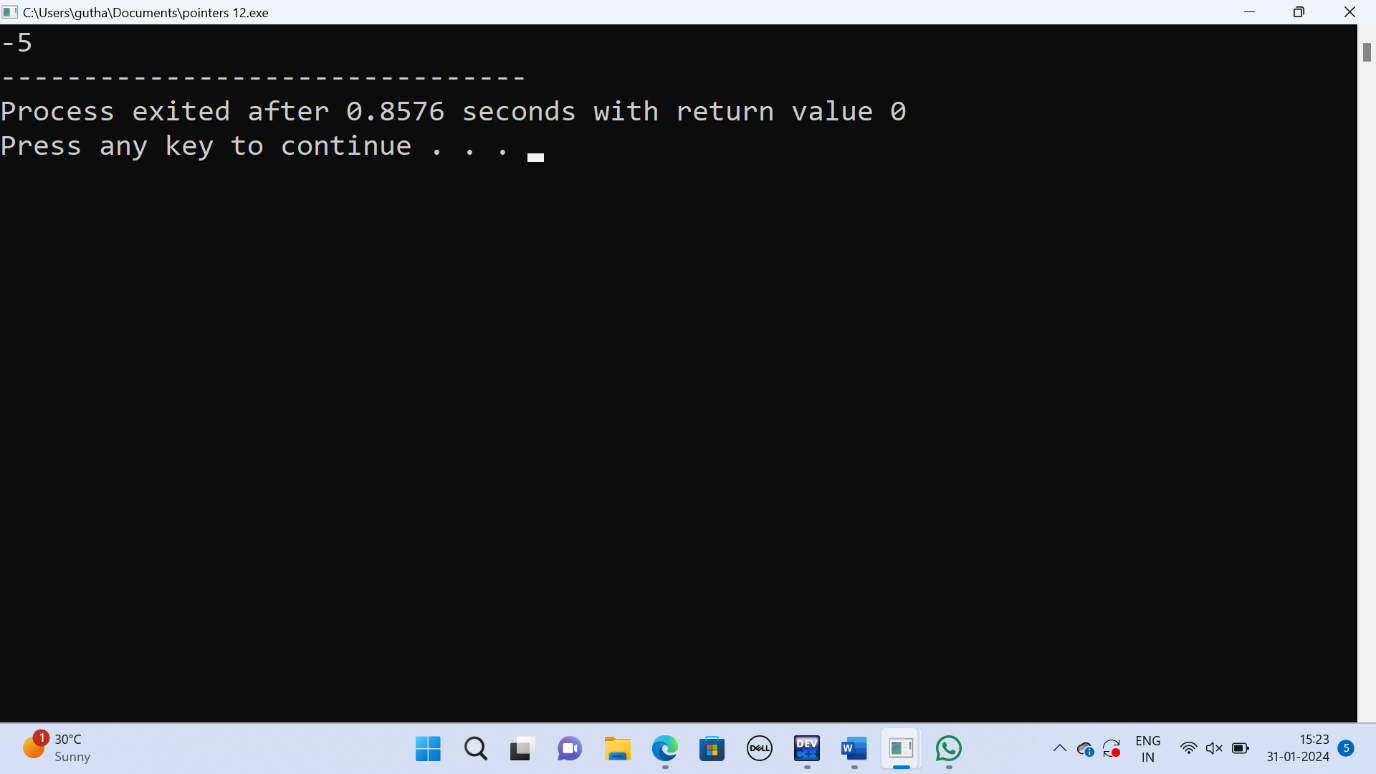
f2(&b, &c);

printf("%d", c - a - b);

return 0;

}

Output:



14.

#include <stdio.h>

struct {

short s[5];

union {

float y;

long z;

} u;

} t;

int main() {

t.s[0] = 1;

t.u.y = 3.14f;

t.u.z = 1000L;

printf("Short array elements: %d\n", t.s[0]);

printf("Union float value: %f\n", t.u.y);

printf("Union long value: %ld\n", t.u.z);

return 0;

}

Output:

