1. #include<stdio.h>

int getLength(char\*p)

{

int count = 0;

while (\*p != '\0')

{

count++;

p++;

}

return count;

}

int main()

{

char str[20] = "OpenGenus";

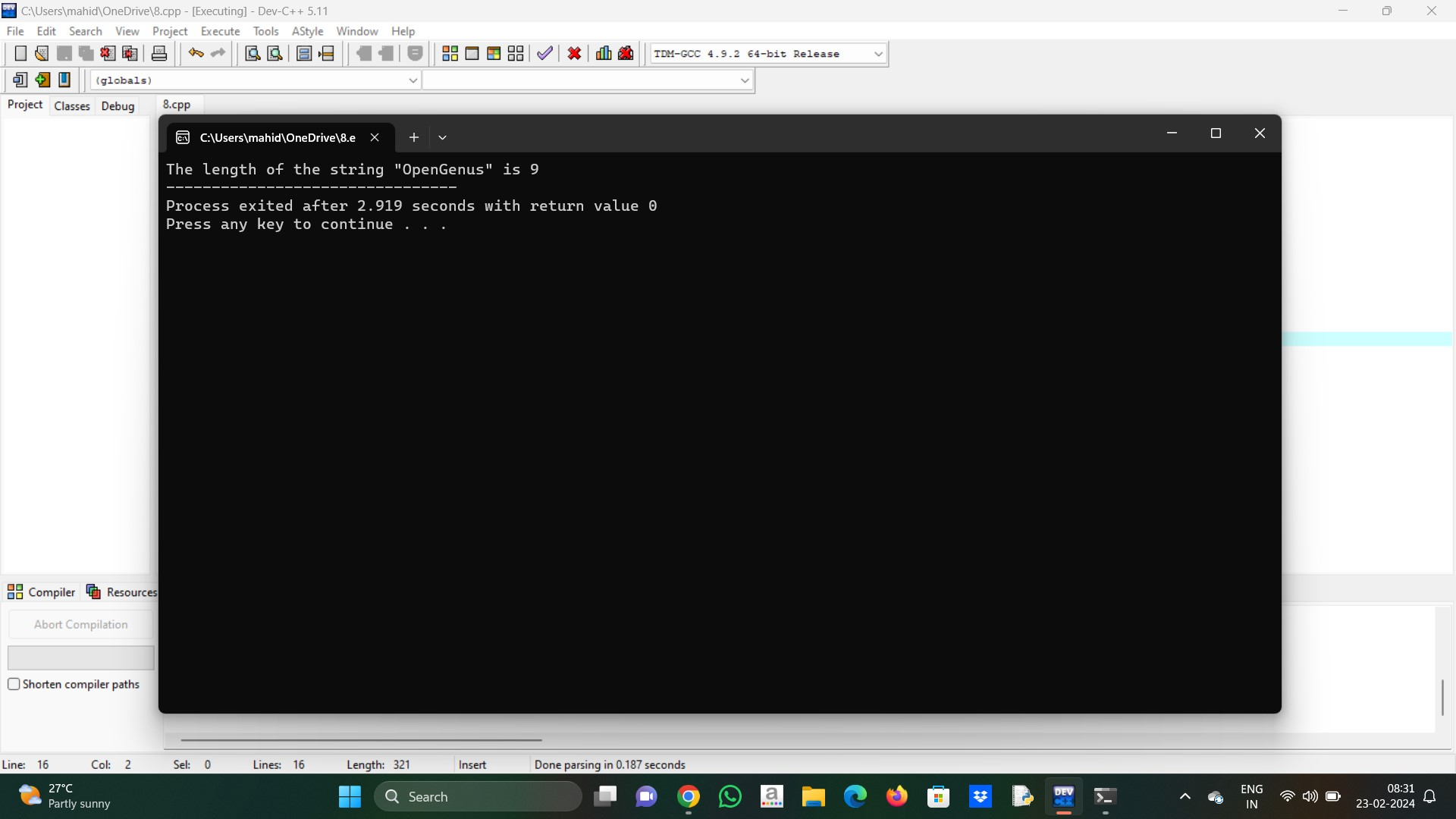
int length = 0;

length = getLength(str);

printf("The length of the string \"%s\" is %d", str, length);

return 0;

}



2. #include <stdio.h>

#include <string.h>

int main()

{

char str[40];

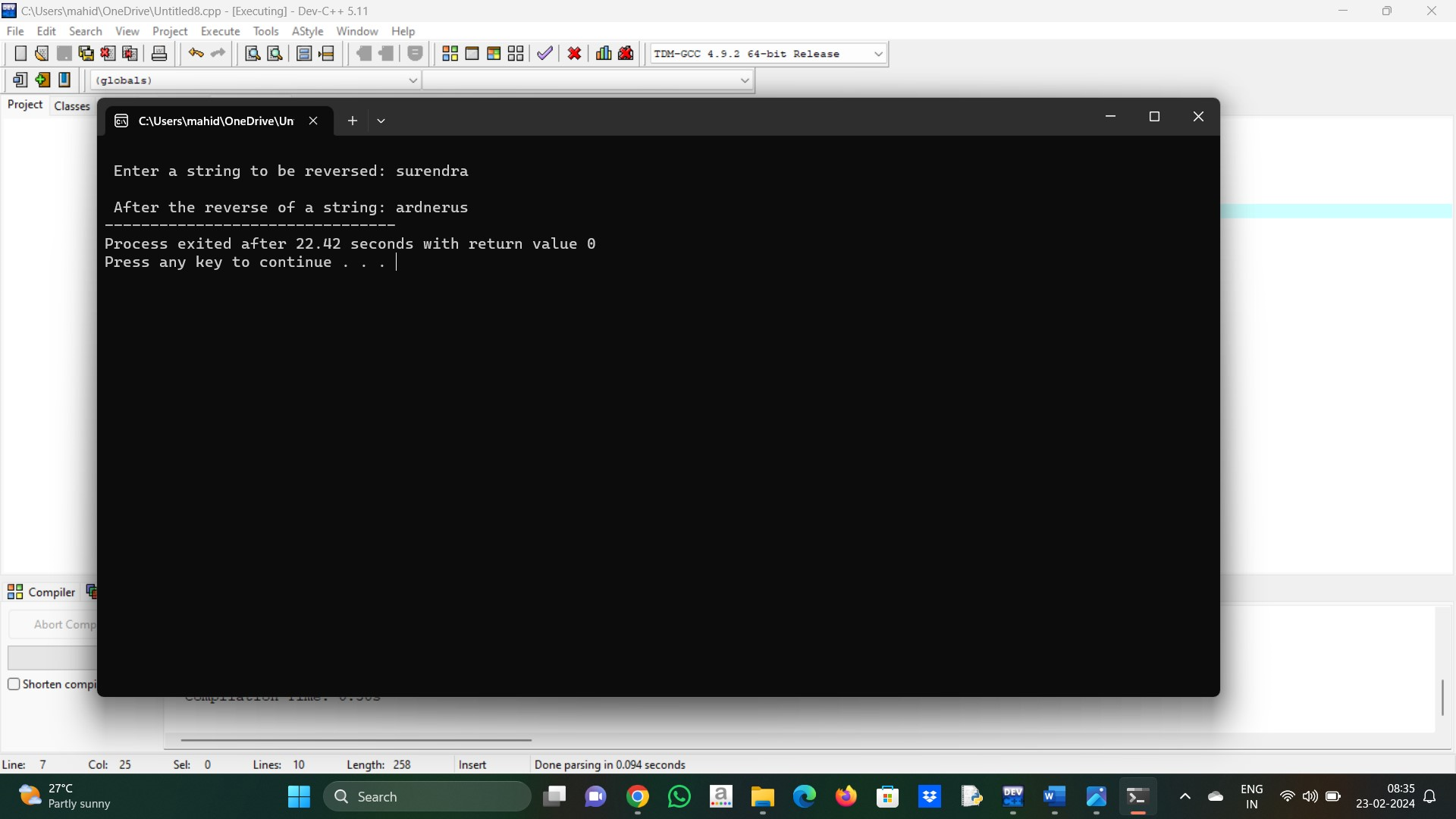
printf (" \n Enter a string to be reversed: ");

scanf ("%s", str);

printf (" \n After the reverse of a string: %s ", strrev(str));

return 0;

}



3. #include<stdio.h>

#include <string.h>

int main()

{

char s[1000],c;

int i,count=0;

printf("Enter the string : ");

gets(s);

printf("Enter character to be searched: ");

c=getchar();

for(i=0;s[i];i++)

{

if(s[i]==c)

{

count++;

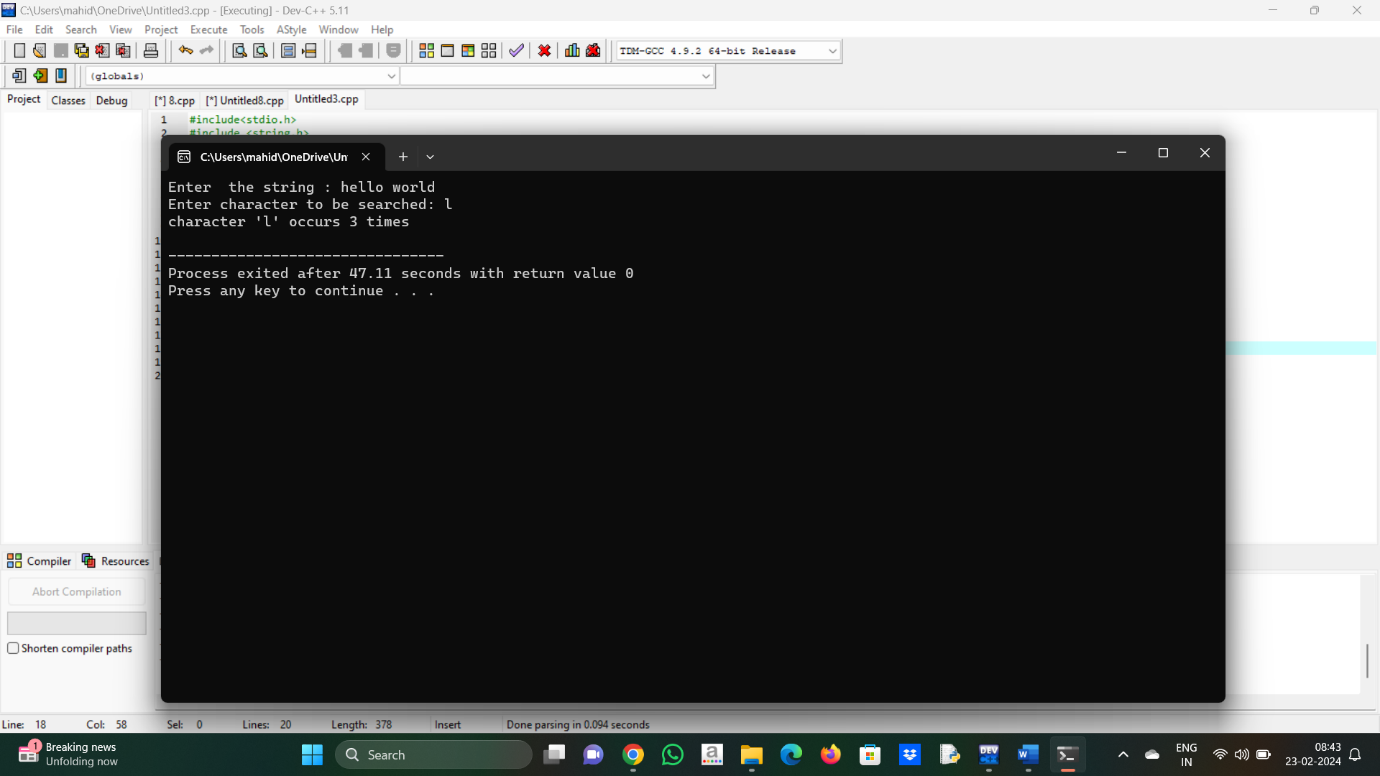
}

}

printf("character '%c' occurs %d times \n ",c,count);

return 0;

}



4. #include <stdio.h>

int main()

{

int a, b, temp;

int \*ptr1, \*ptr2;

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

scanf("%d %d", &a, &b);

printf("\nBefore swapping a = %d and b = %d", a, b);

ptr1 = &a;

ptr2 = &b;

temp = \*ptr1;

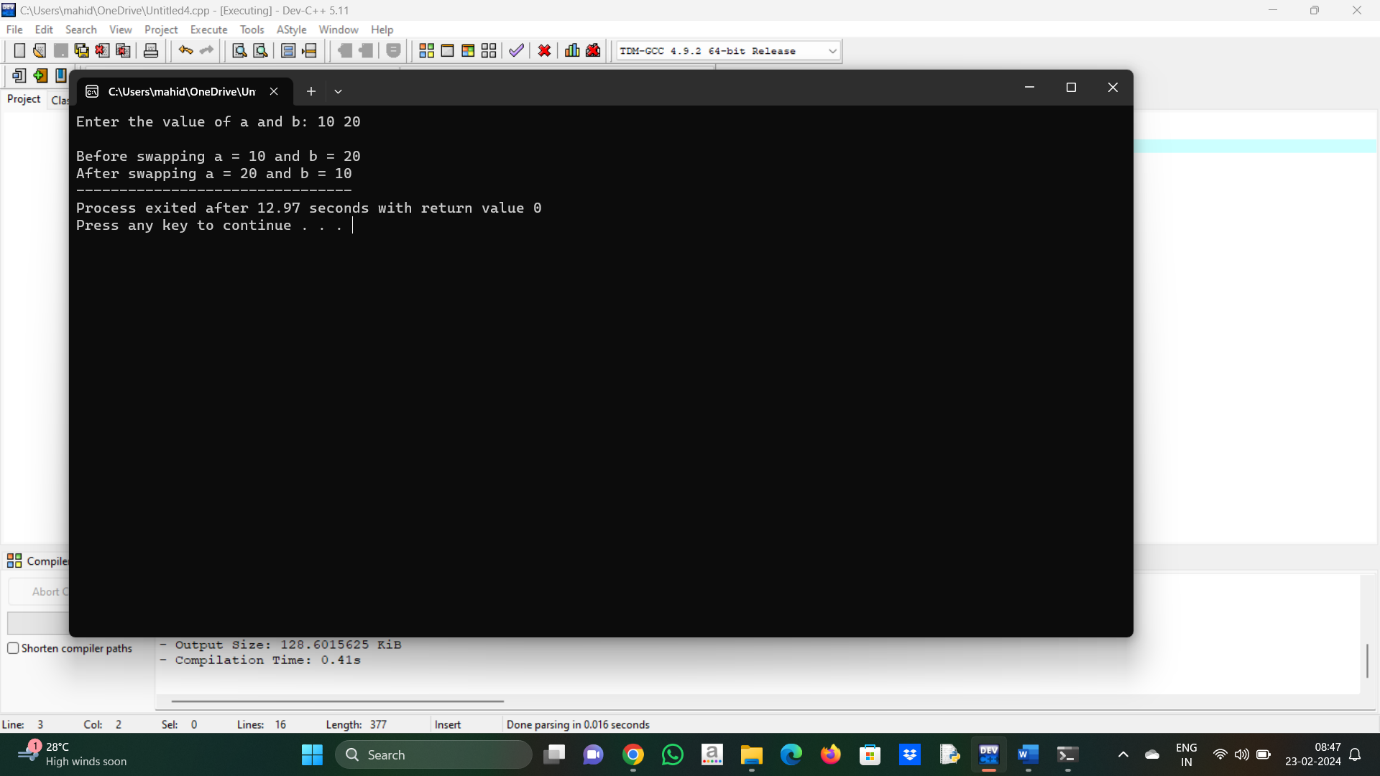
\*ptr1 = \*ptr2;

\*ptr2 = temp;

printf("\nAfter swapping a = %d and b = %d", a, b);

return 0;

}



5. #include<stdio.h>

void sort\_array\_0\_1\_2 ( int A[ ] , int n )

{

int i = 0 , countzero = 0 , countone = 0 , counttwo = 0 ;

while ( i < n )

{

if ( A [ i ] == 0 )

{

countzero = countzero + 1 ;

}

else if ( A [ i ] == 1 )

{

countone = countone + 1 ;

}

else

{

counttwo = counttwo + 2 ;

}

i = i + 1 ;

}

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

{

A [ i ] = 0 ;

}

for ( i = countzero ; i < countzero + countone ; i++ )

{

A [ i ] = 1 ;

}

for ( i = countzero + countone ; i < n ; i++ )

{

A [ i ] = 2 ;

}

}

int main()

{

int i , A[ 100 ] , n;

printf ( " Enter number of elements present in the array: " ) ;

scanf ( " %d " , &n) ;

printf ( " Enter array: \n " ) ;

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

{

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

}

printf ( " Array after sorting : \n " ) ;

sort\_array\_0\_1\_2 ( A , n ) ;

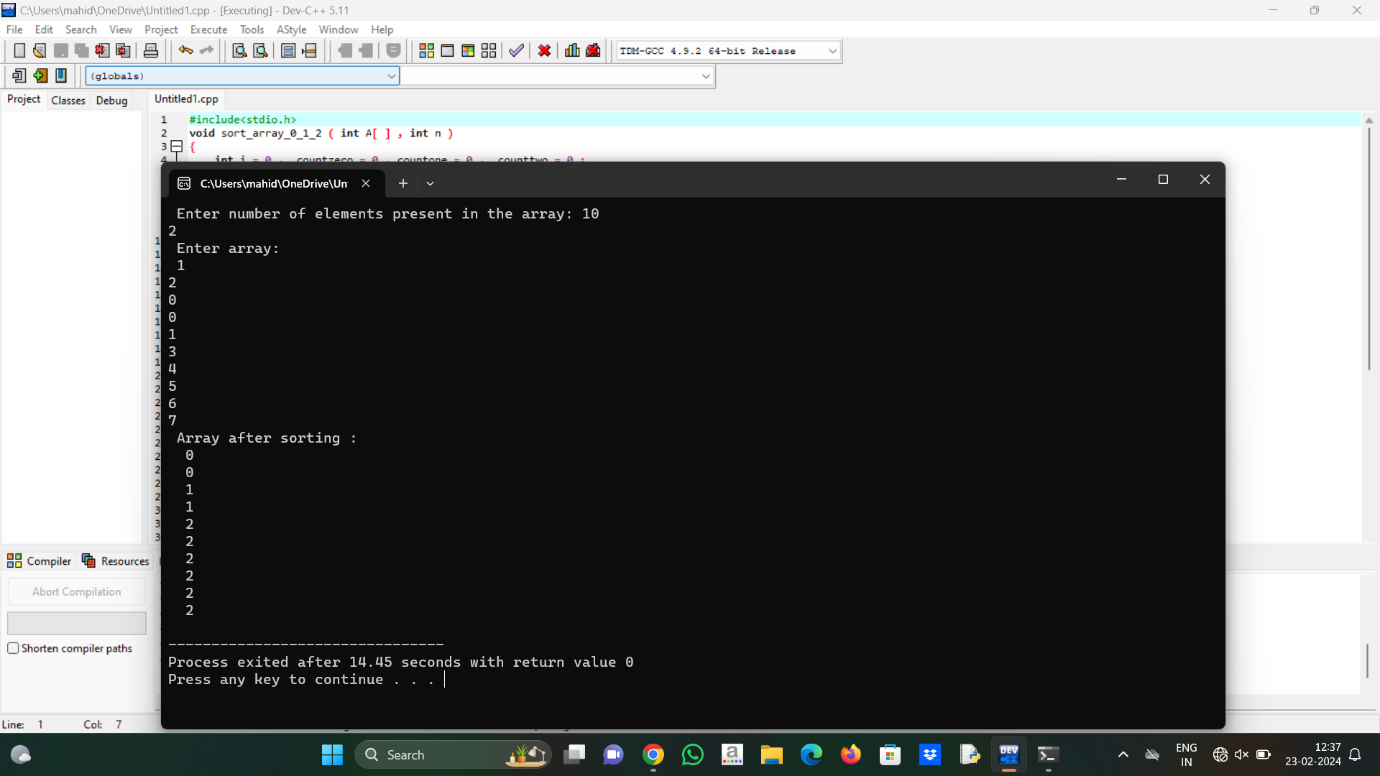
for ( i = 0 ; i < n ; i++ )

{

printf ( " %d \n " , A [ i ] ) ;

}

}



6. #include <stdio.h>

#include <stdlib.h>

int Solution(int arr[], int n)

{

if (n % 2 == 0)

{

int z = n / 2;

int e = arr[z];

int q = arr[z - 1];

int ans = (e + q) / 2;

return ans;

}

else

{

int z = n / 2;

return arr[z];

}

}

int main()

{

int arr1[] = {-5, 3, 6, 12, 15};

int arr2[] = {-12, -10, -6, -3, 4, 10};

int i = sizeof(arr1) / sizeof(arr1[0]);

int j = sizeof(arr2) / sizeof(arr2[0]);

int l = i + j;

int\* arr3 = (int\*)malloc(l \* sizeof(int));

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

{

arr3[k] = arr1[k];

}

int a = 0;

for (int k = i; k < l; k++)

{

arr3[k] = arr2[a++];

}

for (int m = 0; m < l - 1; m++)

{

for (int n = 0; n < l - m - 1; n++)

{

if (arr3[n] > arr3[n + 1])

{

int temp = arr3[n];

arr3[n] = arr3[n + 1];

arr3[n + 1] = temp;

}

}

}

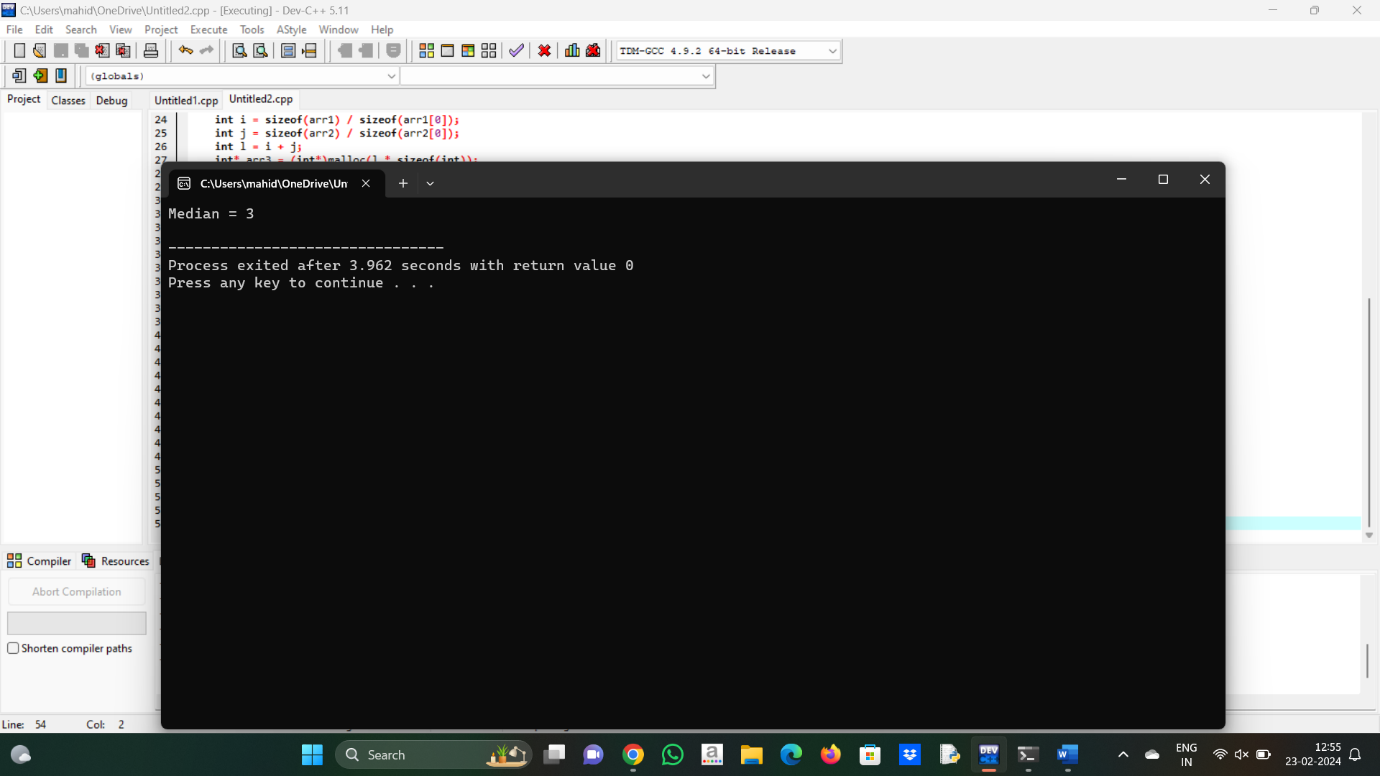
int median = Solution(arr3, l);

printf("Median = %d\n", median);

free(arr3);

return 0;

}



7. #include <stdio.h>

int find\_anagram(char [], char []);

int main()

{

char array1[100], array2[100];

int flag;

printf("Enter the string\n");

gets(array1);

printf("Enter another string\n");

gets(array2);

flag = find\_anagram(array1, array2);

if (flag == 1)

printf("%s and %s are anagrams.\n", array1, array2);

else

printf(" %s and %s are not anagrams.\n", array1, array2);

return 0;

}

int find\_anagram(char array1[], char array2[])

{

int num1[26] = {0}, num2[26] = {0}, i = 0;

while (array1[i] != '\0')

{

num1[array1[i] - 'a']++;

i++;

}

i = 0;

while (array2[i] != '\0')

{

num2[array2[i] -'a']++;

i++;

}

for (i = 0; i < 26; i++)

{

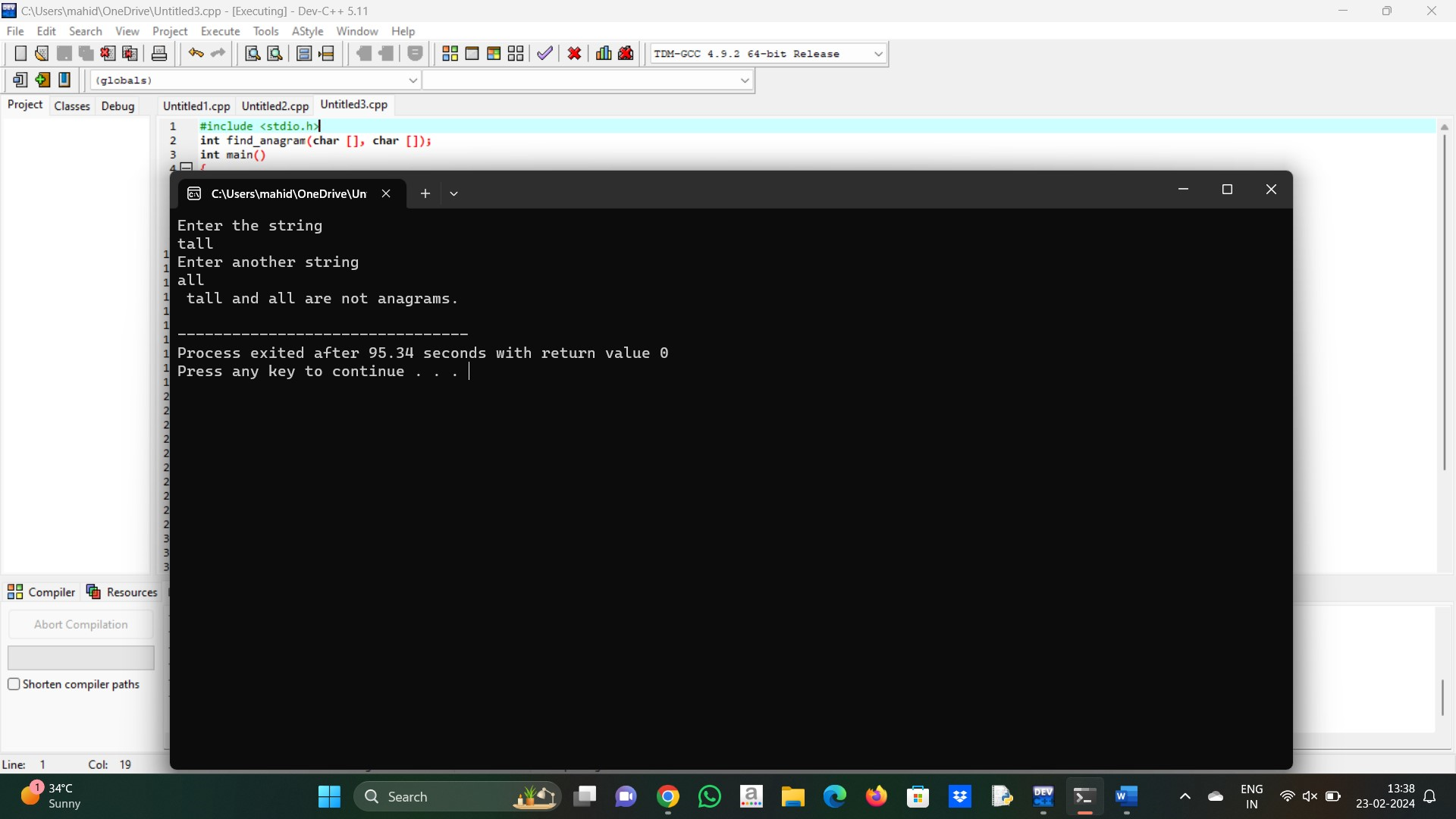
if (num1[i] != num2[i])

return 0;

}

return 1;

}



8. #include <stdbool.h>

#include <stdio.h>

#include <string.h>

int max(int num1, int num2)

{

return (num1 > num2) ? num1 : num2;

}

bool areDistinct(char str[], int i, int j)

{

bool visited[256];

for (int i = 0; i < 256; i++)

visited[i] = 0;

for (int k = i; k <= j; k++) {

if (visited[str[k]] == true)

return false;

visited[str[k]] = true;

}

return true;

}

int longestUniqueSubsttr(char str[])

{

int n = strlen(str);

int res = 0;

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

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

if (areDistinct(str, i, j))

res = max(res, j - i + 1);

return res;

}

int main()

{

char str[] = "surendrareddy";

printf("The input string is %s \n", str);

int len = longestUniqueSubsttr(str);

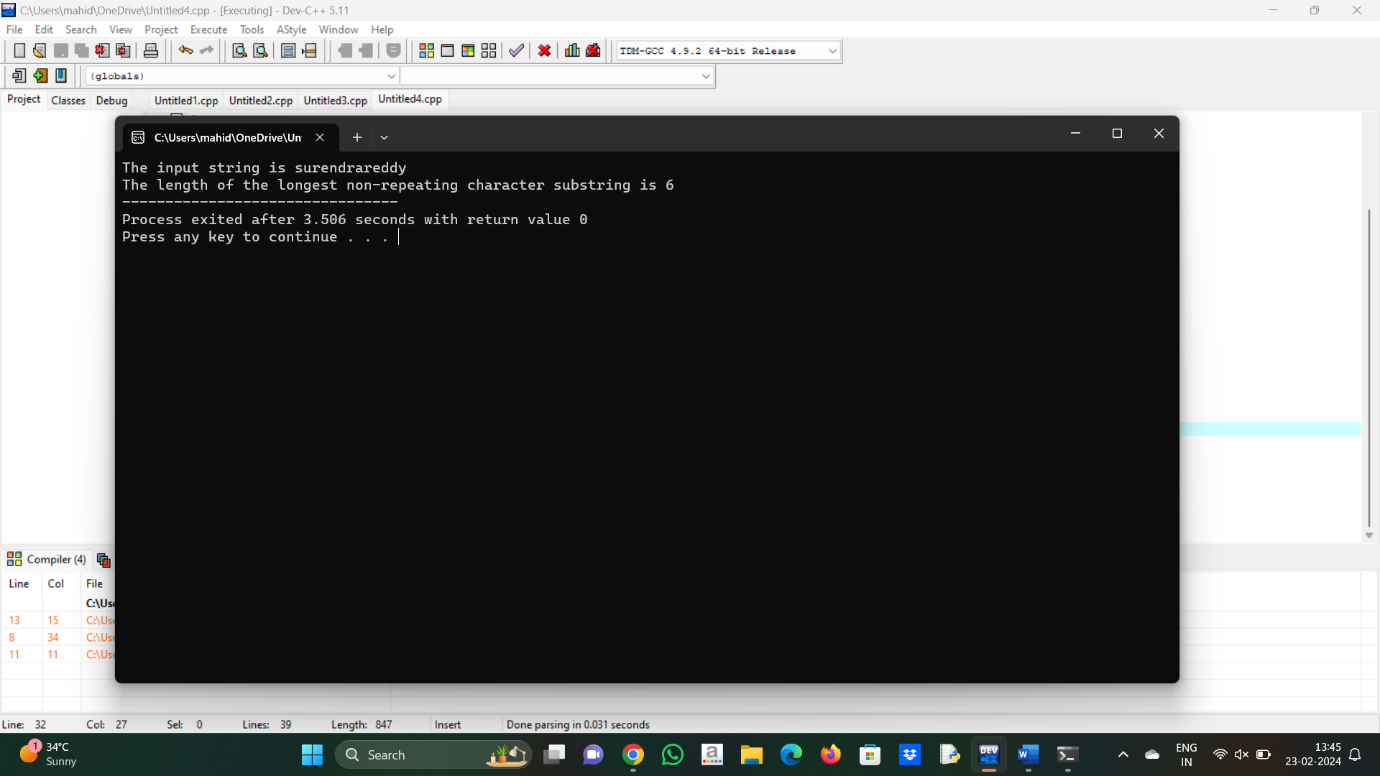
printf("The length of the longest non-repeating "

"character substring is %d",

len);

return 0;

}



9. #include <stdio.h>

#include <stdlib.h>

int cmpfunc(const void\* a, const void\* b)

{

return (\*(int\*)a - \*(int\*)b);

}

int kthSmallest(int arr[], int N, int K)

{

qsort(arr, N, sizeof(int), cmpfunc);

return arr[K - 1];

}

int main()

{

int arr[] = { 12, 3, 5, 7, 19 };

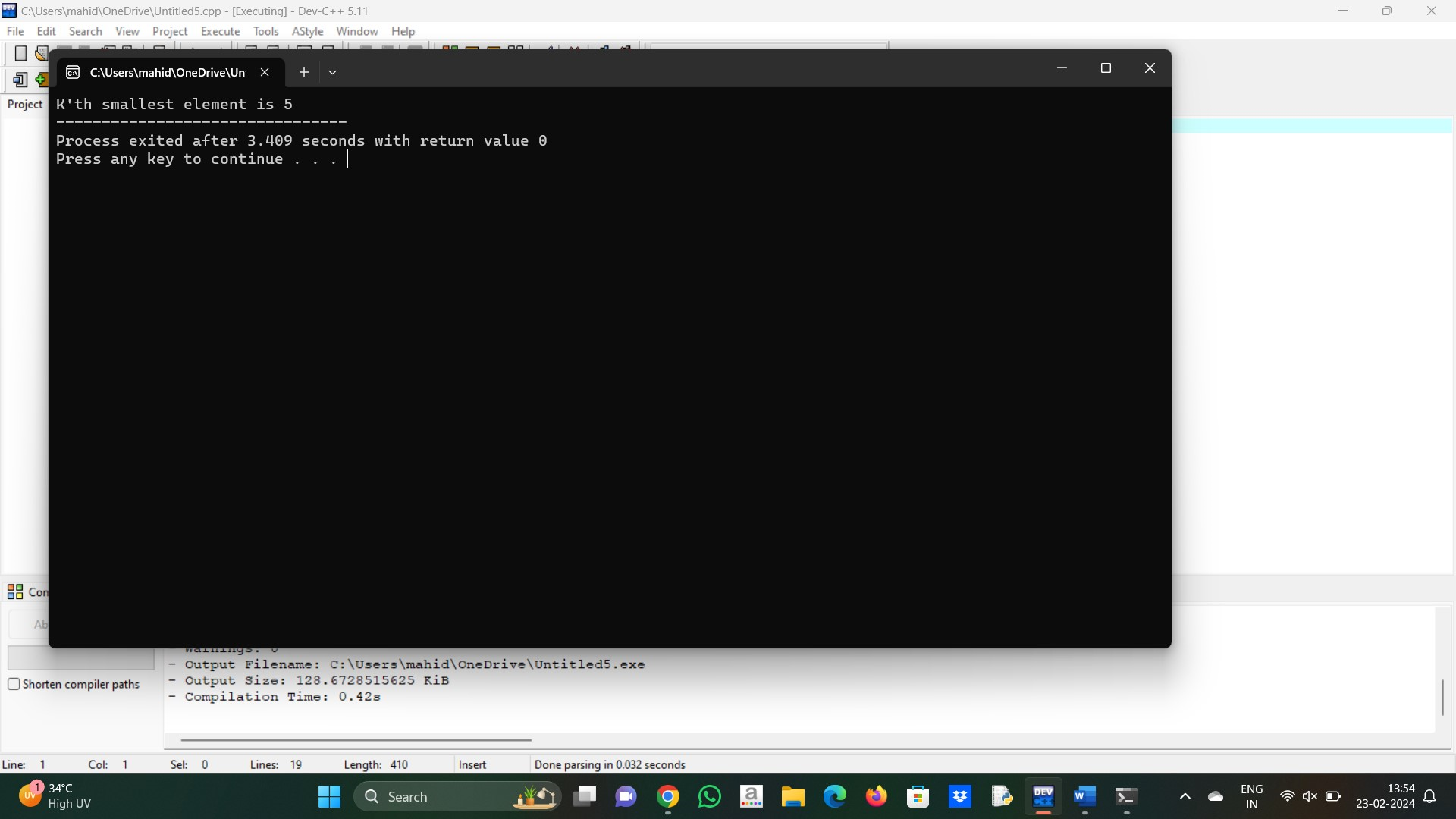
int N = sizeof(arr) / sizeof(arr[0]), K = 2;

printf("K'th smallest element is %d",

kthSmallest(arr, N, K));

return 0;

}



10. #include <stdio.h>

int main() {

char line[150];

int vowels, consonant, digit, space;

vowels = consonant = digit = space = 0;

printf("Enter a line of string: ");

fgets(line, sizeof(line), stdin);

for (int i = 0; line[i] != '\0'; ++i) {

line[i] = (line[i]);

if (line[i] == 'a' || line[i] == 'e' || line[i] == 'i' ||

line[i] == 'o' || line[i] == 'u') {

++vowels;

}

else if ((line[i] >= 'a' && line[i] <= 'z')) {

++consonant;

}

else if (line[i] >= '0' && line[i] <= '9') {

++digit;

}

else if (line[i] == ' ') {

++space;

}

}

printf("Vowels: %d", vowels);

printf("\nConsonants: %d", consonant);

printf("\nDigits: %d", digit);

printf("\nWhite spaces: %d", space);

return 0;

}

