**Name: Swaranjana Nayak**

**Reg. No.: 19BCE0977**

**Date: 05/08/2020**

**PART II : ARRAYS**

1. **WAP to sort n alphabets (both upper and lower) in an array in ascending order.**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#define N 101

// C program to sort an array of alphabets (both upper and lower) in an array in ascending order

int main()

{

char str[N], temp, c1, c2;

int i, j, n;

printf("----Program to sort an array of n alphabets----\n\n");

printf("Enter n: ");

scanf("%d", &n);

fflush(stdin);

printf("Enter the string: ");

fgets(str, N-1, stdin);

fflush(stdin);

// Input validation

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

{

if(isalpha(str[i]) == 0 || str[i] == '\n' || str[i] == '\0')

{

printf("Invalid input!");

printf("\nPlease enter all alphabets without separation: ");

fgets(str, N-1, stdin);

}

}

// Sorting when case sensitive

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

{

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

{

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

{

temp = str[i];

str[i] = str[j];

str[j] = temp;

}

}

}

printf("[CASE SENSITIVE]\n");

printf("{");

for(i = 0; i< n-1; i++)

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

printf("%c}\n", str[n-1]);

// NON CASE SENSITIVE SORTING

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

{

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

{

c1 = tolower(str[i]);

c2 = tolower(str[j]);

if(c2 > c1)

{

temp = str[i];

str[i] = str[j];

str[j] = temp;

}

}

}

printf("\n[NOT CASE SENSITIVE]\n");

printf("{");

for(i = 0; i< n-1; i++)

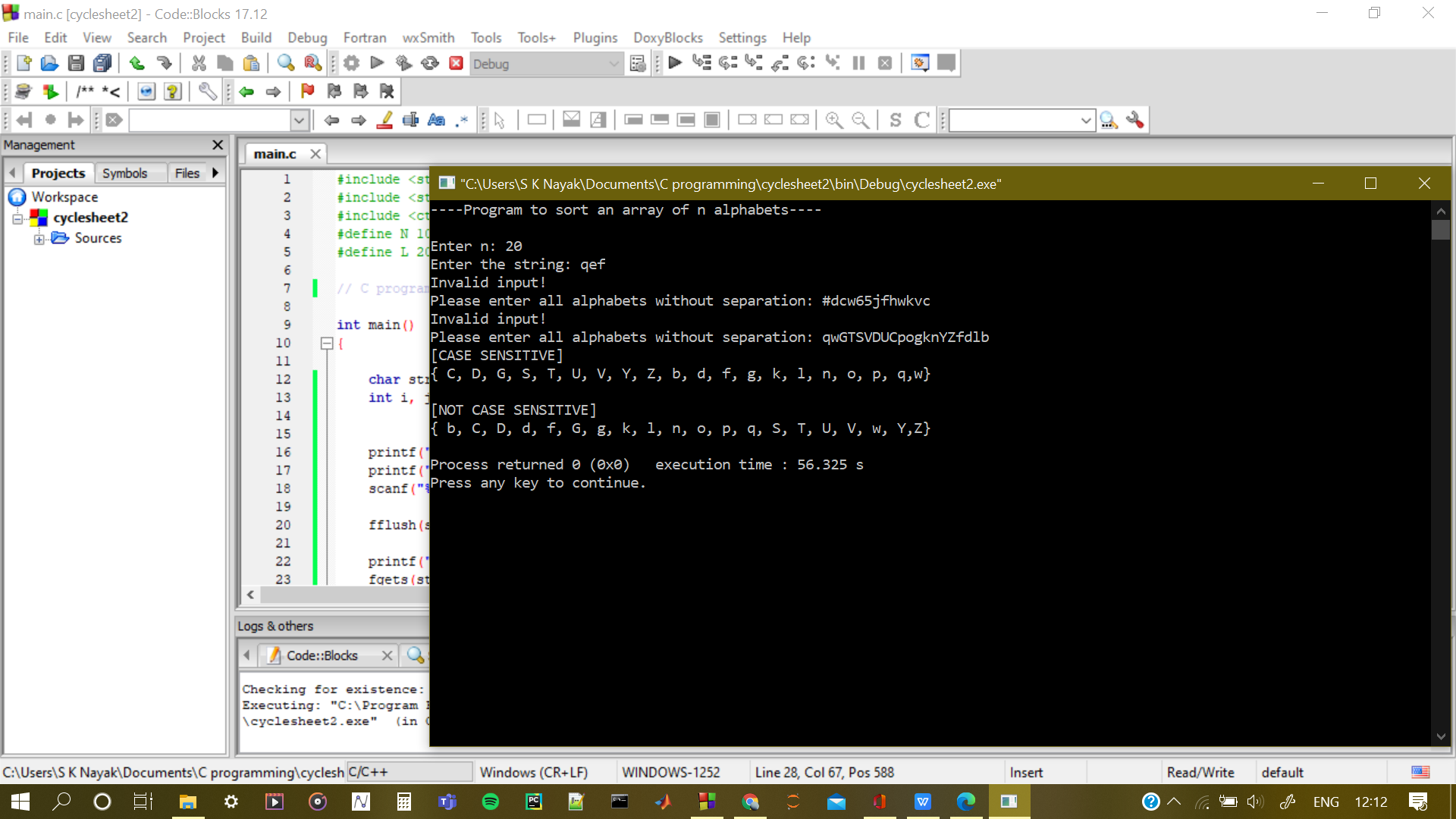
printf(" %c,", str[i]);

printf("%c}\n", str[n-1]);

return 0;

}

**Output:**



1. **WAP to extract odd, even and prime numbers in an array to 3 different arrays and display this array with proper formatting**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#define N 100

// C program to extract odd, even and prime numbers in an array to 3 different arrays and display this array

int parity(int a);

int primality(int a);

int main()

{

int num[N], odd[N], even[N], prime[N];

int i, jo = 0, je = 0, jp = 0, n, status, flag;

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

scanf("%d", &n);

fflush(stdin);

// Taking input + validation

do {printf("Enter the array: ");

flag = 1;

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

{

status = scanf("%d", &num[i]);

if(status == 0 || num[i] < 0)

{

flag = 0;

}

}

if(flag == 0)

{

printf("Invalid array!\n");

fflush(stdin);

}

}while(flag == 0);

// Defining the arrays

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

{

if(parity(num[i]) == 0)

{

even[je] = num[i];

je++;

}

if(parity(num[i]) == 1)

{

odd[jo] = num[i];

jo++;

}

if(primality(num[i]) == 1)

{

prime[jp] = num[i];

jp++;

}

}

// Printing the arrays

printf("\n--- EVEN NUMBER ARRAY ---\n");

printf("{");

for(i = 0; i < je-1; i++)

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

printf(" %d}\n", even[je-1]);

printf("\n--- ODD NUMBER ARRAY ---\n");

printf("{");

for(i = 0; i < jo-1; i++)

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

printf(" %d}\n", odd[jo-1]);

printf("\n--- PRIME NUMBER ARRAY ---\n");

printf("{");

for(i = 0; i < jp-1; i++)

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

printf(" %d}\n", prime[jp-1]);

return 0;

}

// Function to check odd or even

int parity(int a)

{

if(a % 2 == 0)

{

return 0;

}

else

{

return 1;

}

}

// Function to check prime or not

int primality(int a)

{

int i;

if(a <= 1)

{

return 0;

}

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

{

if(a%i == 0)

{

return 0;

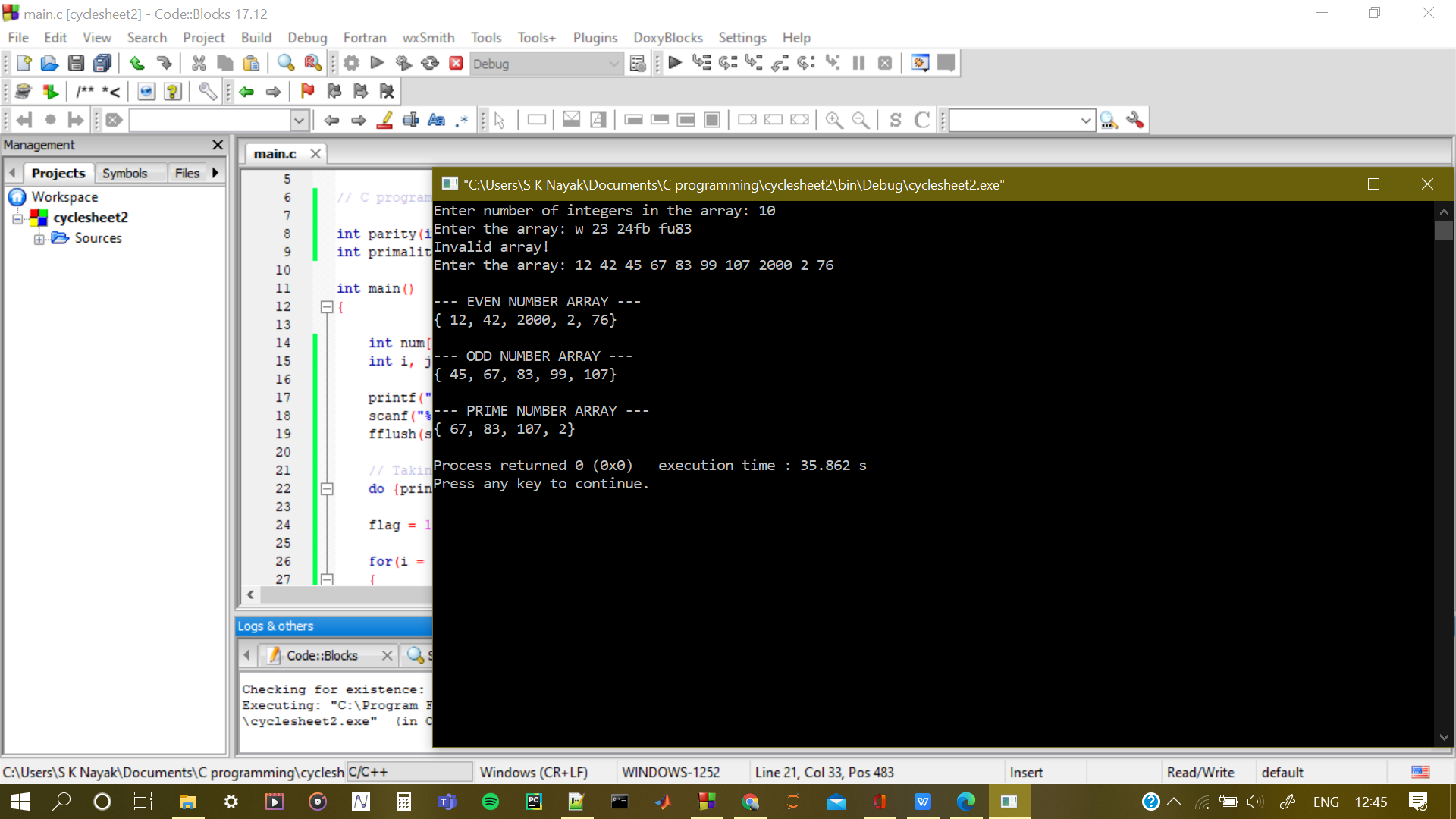
}

}

return 1;

}

**Output:**



1. **WAP to rearrange positive and negative integers in an 1-D array**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#define N 100

// C program to sort an integer array

void insert(char a[N], char e, int p, int l);

int main()

{

int num[N], temp;

int i, j, n, status, flag;

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

scanf("%d", &n);

fflush(stdin);

// Taking input + validation

do {printf("Enter the array: ");

flag = 1;

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

{

status = scanf("%d", &num[i]);

if(status == 0)

{

flag = 0;

break;

}

}

if(flag == 0)

{

printf("Invalid array!\n");

fflush(stdin);

}

}while(flag == 0);

// Sorting the array

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

{

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

{

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

{

temp = num[i];

num[i] = num[j];

num[j] = temp;

}

}

}

// Printing the resulting array

printf("\n\n--- RESULTANT ARRAY ---\n");

printf("{");

for(i = 0; i < n-1; i++)

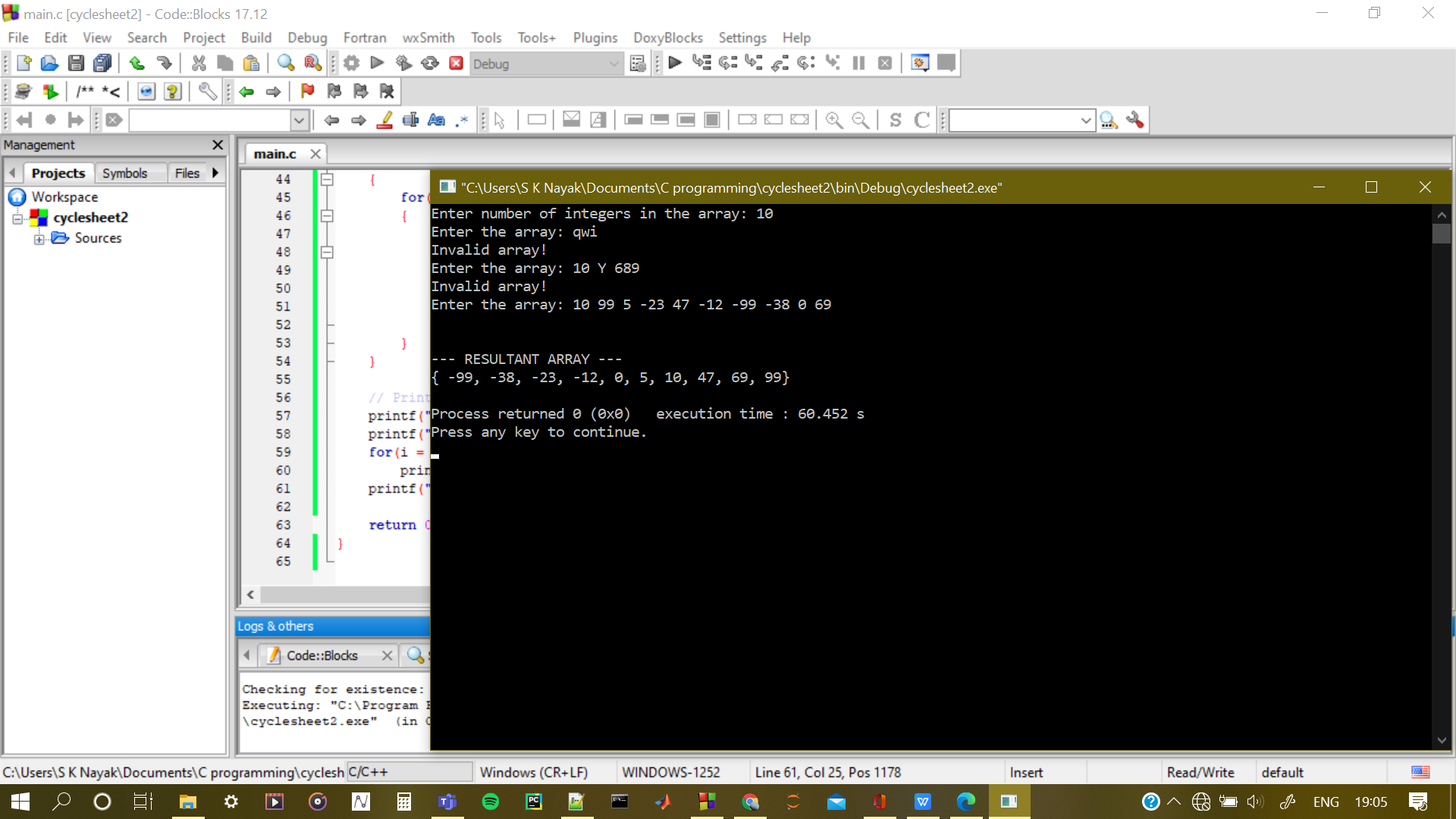
printf(" %d,", num[i]);

printf(" %d}\n", num[n-1]);

return 0;

}

**Output:**



1. **WAP to insert an element into the array, considering all the 3 cases i.e., ** 
   1. **Beginning of the array **
   2. **Middle of the array **
   3. **End of the array**

**Finally display the resultant array**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#define N 100

// C program to insert element into an array at chosen position

void insert(char a[N], char e, int p, int l);

int main()

{

char arr[N], elem;

int i, n, status, flag, begin = 0, end, middle, option;

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

scanf("%d", &n);

fflush(stdin);

end = n-1;

middle = (begin + end)/2;

// Taking input + validation

do {printf("Enter the array (no spaces): ");

flag = 1;

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

{

status = scanf("%c", &arr[i]);

if(status == 0 || arr[i] == '\n')

{

flag = 0;

}

}

if(flag == 0)

{

printf("Invalid array!\n");

fflush(stdin);

}

}while(flag == 0);

// ELEMENT TO BE ENTERED - USER INPUT

printf("Please enter element you want to enter: ");

fflush(stdin);

scanf("%c", &elem);

// Choose where you want to enter:

printf("\nPLEASE CHOOSE WHERE YOU WANT TO INSERT THE ELEMENT FROM THE GIVEN OPTIONS:\n");

printf("1. Beginning\n2. Middle\n3. End\n\n");

do

{

printf("Enter the option number: ");

fflush(stdin);

status = scanf("%d", &option);

flag = 1;

if(status == 0)

{

printf("Invalid input!\n");

flag = 0;

option = 0;

}

switch(option)

{

case 1:

insert(arr, elem, begin, n);

break;

case 2:

insert(arr, elem, middle, n);

break;

case 3:

insert(arr, elem, end, n);

break;

default:

flag = 0;

break;

}

}while(flag == 0);

// Printing the resulting array

printf("\n\n--- RESULTANT ARRAY ---\n");

printf("{");

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

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

printf(" %c}\n", arr[n]);

return 0;

}

// Function to insert an element e in array a of length l at position p

void insert(char a[N], char e, int p, int l)

{

int i;

for(i = l; i > p; i--)

{

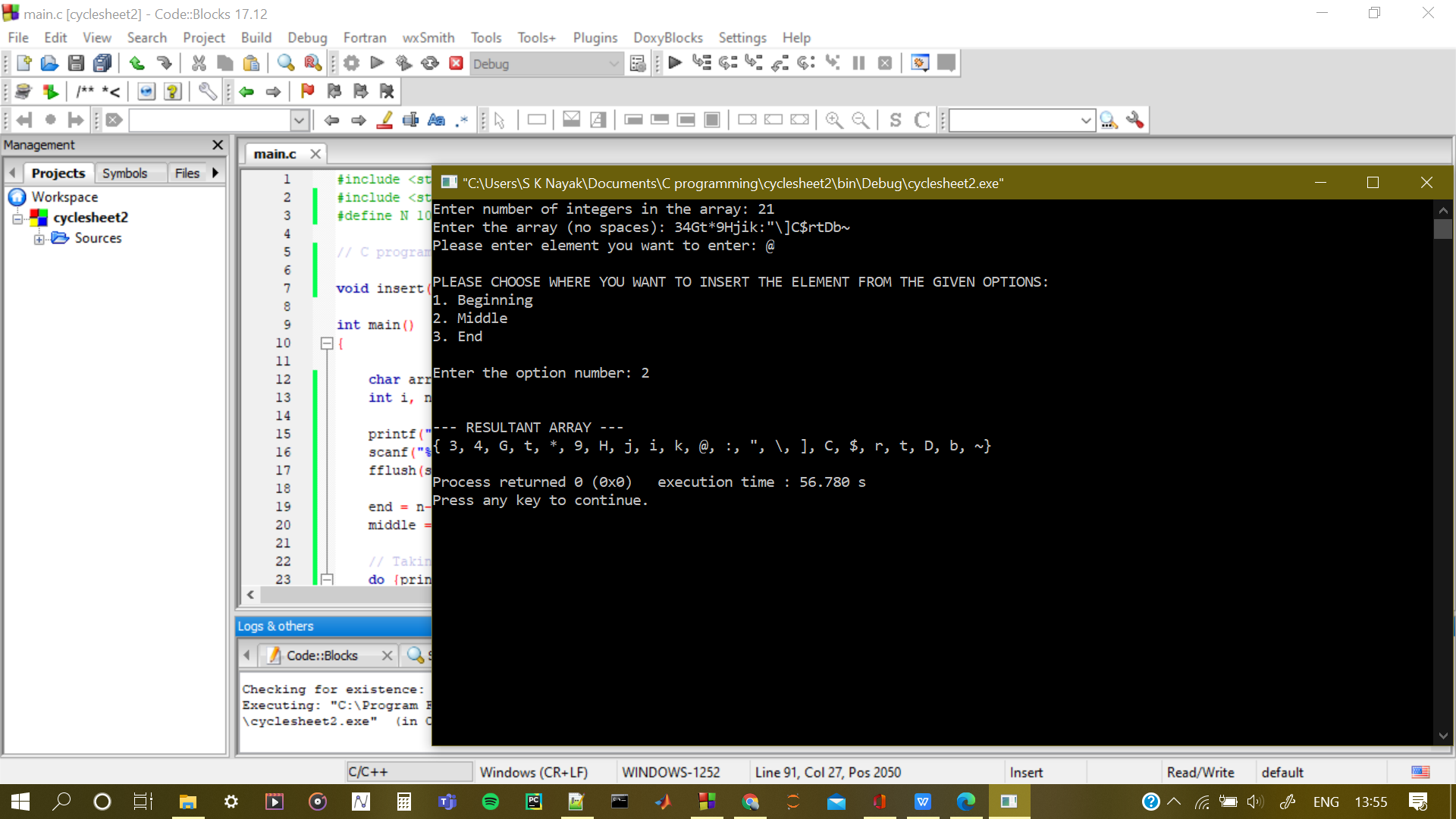
a[i] = a[i-1];

}

a[p] = e;

}

**Output:**



1. **WAP a program to swap adjacent elements in a one dimensional array**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#define N 100

// C program to swap adjacent elements an array at chosen position

void insert(char a[N], char e, int p, int l);

int main()

{

char arr[N], temp;

int i, n, status, flag, position, option;

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

scanf("%d", &n);

fflush(stdin);

// Taking input + validation

do {printf("Enter the array (no spaces): ");

flag = 1;

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

{

status = scanf("%c", &arr[i]);

if(status == 0 || arr[i] == '\n')

{

flag = 0;

break;

}

}

if(flag == 0)

{

printf("Invalid array!\n");

fflush(stdin);

}

}while(flag == 0);

// ELEMENT TO BE ENTERED - USER INPUT

printf("Please enter position you want swap at: ");

fflush(stdin);

scanf("%d", &position);

if(position > 0 && position < n-1)

{

printf("\nDo you want to swap with:-\n");

printf("1. Element behind the position\n2. Element in front of the position\n\n");

printf("Enter the option: ");

scanf("%d", &option);

while(option != 1 && option != 2)

{

printf("Invalid option!\n");

printf("Please enter a valid option");

fflush(stdin);

scanf("%d", &option);

}

}

// SWAPPING

// Case 1: swap at first position

if(position == 0)

{

temp = arr[0];

arr[0] = arr[1];

arr[1] = temp;

}

else if(position > 0 && position < n-1)

{

temp = arr[position];

if(option == 1)

{

arr[position] = arr[position - 1];

arr[position - 1] = temp;

}

else if(option == 2)

{

arr[position] = arr[position + 1];

arr[position + 1] = temp;

}

}

else

{

temp = arr[position];

arr[position] = arr[position - 1];

arr[position - 1] = temp;

}

// Printing the resulting array

printf("\n\n--- RESULTANT ARRAY ---\n");

printf("{");

for(i = 0; i < n-1; i++)

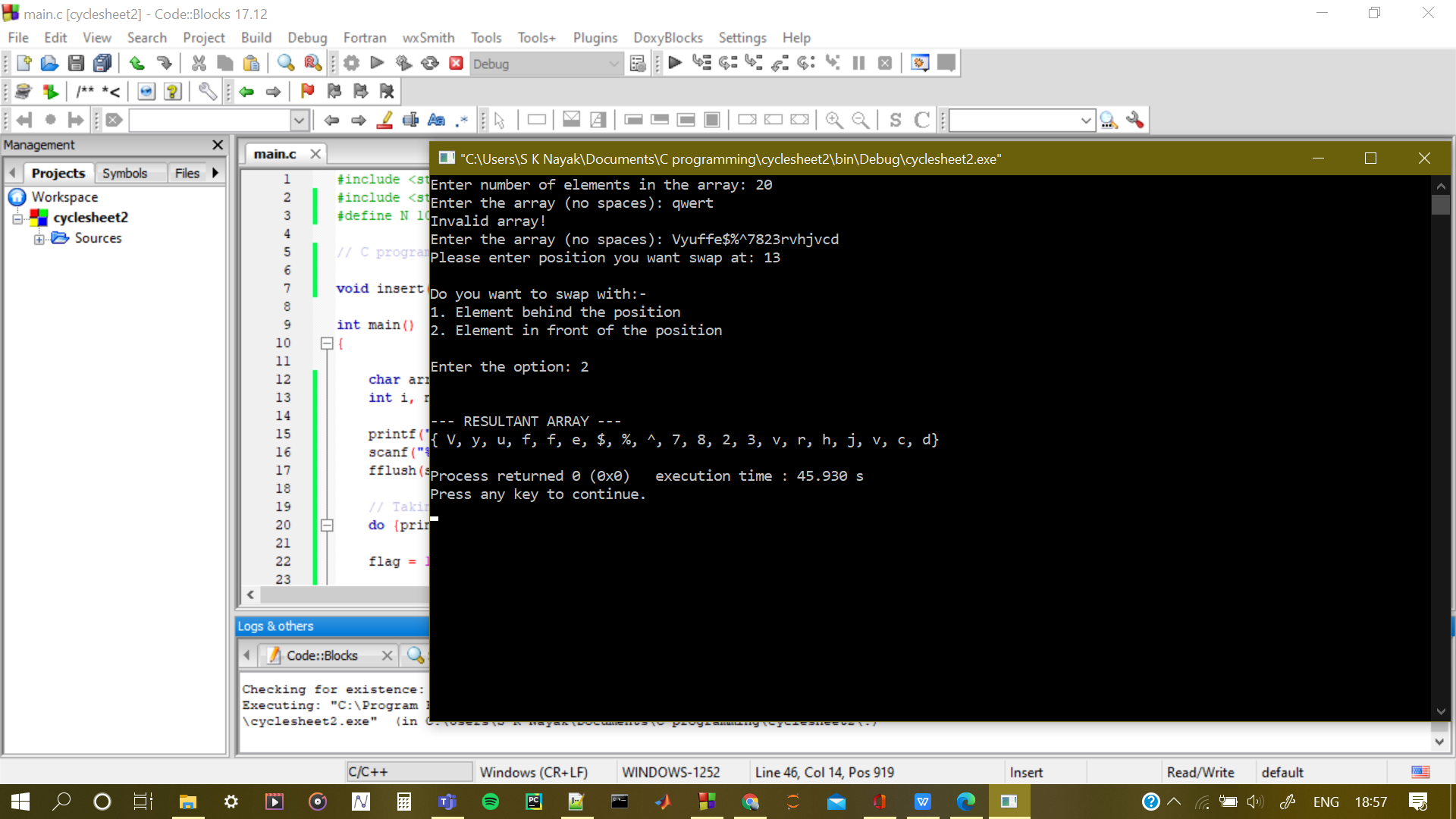
printf(" %c,", arr[i]);

printf(" %c}\n", arr[n-1]);

return 0;

}

**Output:**



1. **WAP to search an element in an array and replace with an element \***

**Code:**

#include <stdio.h>

#include <stdlib.h>

#define N 100

// C program to find and replace an element in an array

void insert(char a[N], char e, int p, int l);

int main()

{

char arr[N], find, replace;

int i, n, status, flag;

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

scanf("%d", &n);

fflush(stdin);

// Taking input + validation

do {printf("Enter the array(no spaces unless it's an element): ");

flag = 1;

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

{

status = scanf("%c", &arr[i]);

if(status == 0 || arr[i] == '\n')

{

flag = 0;

break;

}

}

if(flag == 0)

{

printf("Invalid array!\n");

fflush(stdin);

}

}while(flag == 0);

// The element to be found:

printf("Enter the element you want to find: ");

fflush(stdin);

scanf("%c", &find);

// The element you want to replace it with

printf("Enter the character you want to replace the element with: ");

fflush(stdin);

scanf("%c", &replace);

// Finding and replacing the element

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

{

if(arr[i] == find)

{

arr[i] = replace;

}

}

// Printing the resulting array

printf("\n\n--- RESULTANT ARRAY ---\n");

printf("{");

for(i = 0; i < n-1; i++)

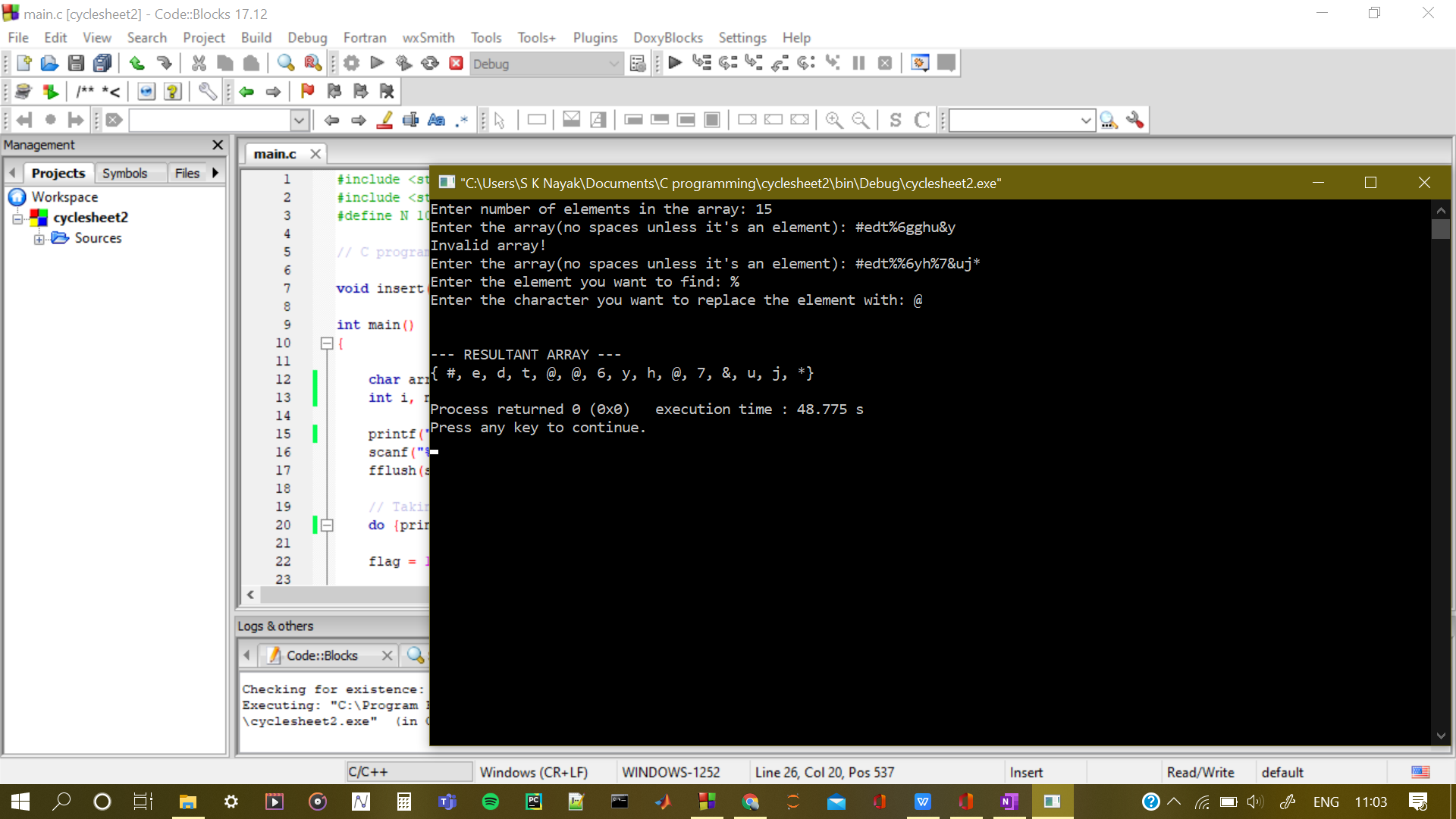
printf(" %c,", arr[i]);

printf(" %c}\n", arr[n-1]);

return 0;

}

**Output:**



1. **WAP to split the given 1-D array into repeating and non-repeating elements of the array**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#define N 100

// C program to find and replace an element in an array

void insert(char a[N], char e, int p, int l);

int main()

{

char arr[N], repeating[N], non\_repeating[N];

int i, j, k, n, r = 0, nr = 0, status, flag, repeats;// temp;

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

scanf("%d", &n);

fflush(stdin);

// Taking input + validation

do {

printf("Enter the array(no spaces unless it's an element): ");

flag = 1;

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

{

status = scanf("%c", &arr[i]);

if(status == 0 || arr[i] == '\n')

{

flag = 0;

break;

}

}

if(flag == 0)

{

printf("Invalid array!\n");

fflush(stdin);

}

}while(flag == 0);

//Starting from last element and moving down is a good idea

// Deleting repeated elem is easy and it won't be revisited

for(i = n-1; i >= 0; i--)

{

// Set default value as 0 each time a new element is evaluated

repeats = 0;

// Evaluating is the element has ever appeared before

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

{

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

{

// Change flag var to 1

// To signal that element is repeating

repeats = 1;

// Remove the repeating element from array

// To avoid repetition in repeating array

for(k = j+1; k <= i; k++)

{

arr[k-1] = arr[k];

}

// Everything is shifter 1 elem back

// So ith position is new too

i--;

}

}

// If repeats is 0 then add element to non repeating array

// If repeats is 1 then add element to repeating array

if(repeats == 0)

{

non\_repeating[nr] = arr[i];

nr++;

}

else

{

repeating[r] = arr[i];

r++;

}

}

// Printing the repeating array

printf("\n\n--- REPEATING ARRAY ---\n");

printf("{");

for(i = 0; i < r-1; i++)

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

printf(" %c}\n", repeating[r-1]);

// Printing the non-repeating array

printf("\n\n--- NON REPEATING ARRAY ---\n");

printf("{");

for(i = 0; i < nr-1; i++)

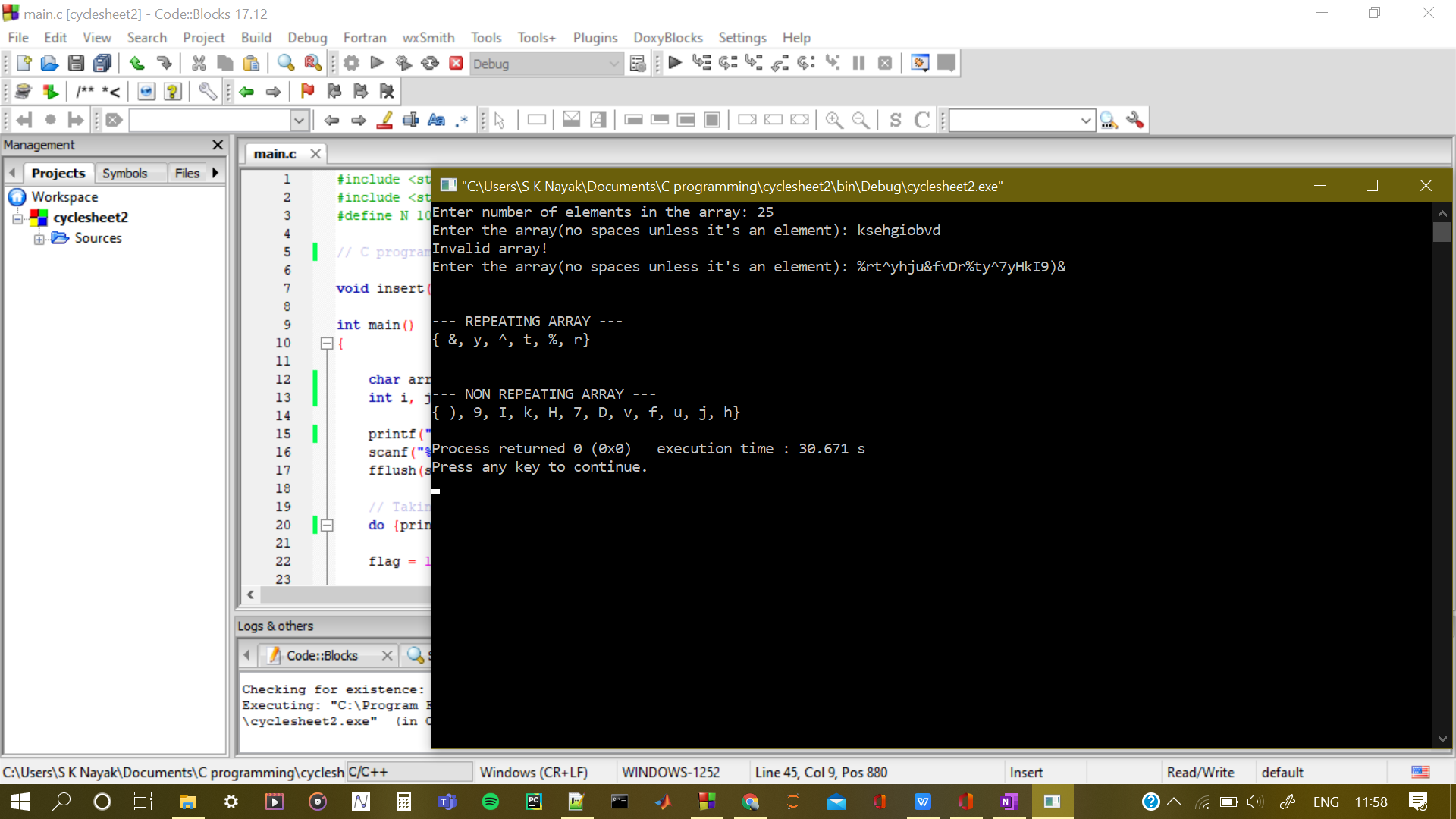
printf(" %c,", non\_repeating[i]);

printf(" %c}\n", non\_repeating[nr-1]);

return 0;

}

**Output:**



1. **WAP to find the sum of two matrix.**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#define N 100

// C program to find sum of two matrices

int main()

{

float a[N][N], b[N][N], c[N][N];

int i, j, m, n, status, flag;

printf("Enter number of rows in the matrices (A, B): ");

scanf("%d", &m);

fflush(stdin);

printf("Enter number of columns in the matrices (A, B): ");

scanf("%d", &n);

fflush(stdin);

printf("\n\n------- MATRIX A -------\n");

// Taking input for matrix A + validation

do {

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

{

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

flag = 1;

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

{

status = scanf("%f", &a[i][j]);

if(status == 0)

{

flag = 0;

break;

}

}

if(flag == 0)

{

printf("Invalid entry!\n");

fflush(stdin);

break;

}

}

}while(flag == 0);

printf("\n\n\n------- MATRIX B -------\n");

// Taking input for matrix B + validation

do {

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

{

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

flag = 1;

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

{

status = scanf("%f", &b[i][j]);

if(status == 0)

{

flag = 0;

break;

}

}

if(flag == 0)

{

printf("Invalid entry!\n");

fflush(stdin);

break;

}

}

}while(flag == 0);

// Adding the matrices

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

{

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

{

c[i][j] = a[i][j] + b[i][j];

}

}

// Printing the matrix C

printf("\n\n------- MATRIX C -------\n");

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

{

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

{

printf(" %0.4f,", c[i][j]);

}

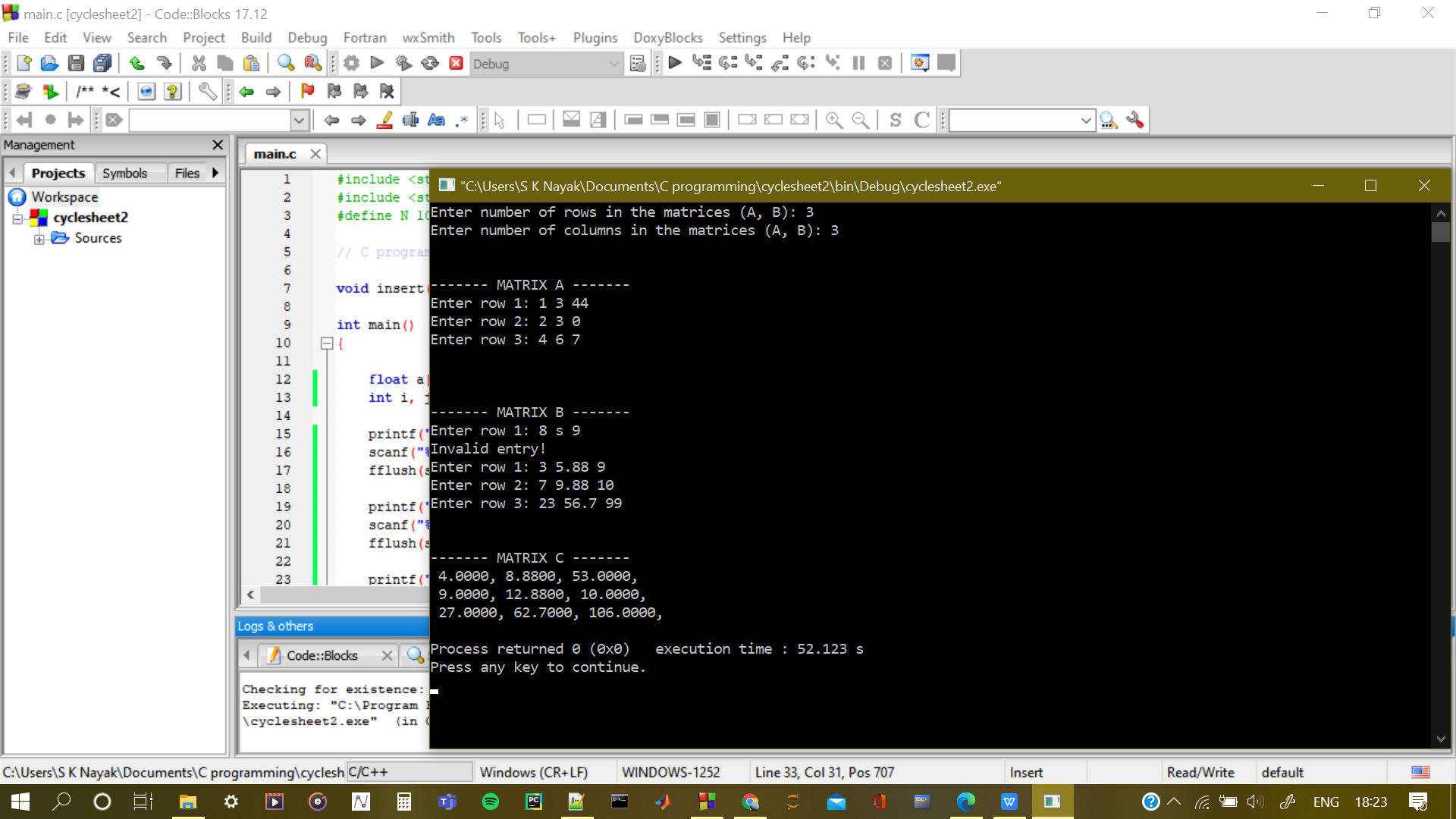
printf("\n");

}

return 0;

}

**Output:**



1. **WAP to transpose the given matrix.**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#define N 100

// C program to find transpose of a matrix

int main()

{

float a[N][N], b[N][N];// c[N][N];

int i, j, m, n, status, flag;

printf("Enter number of rows in the matrix (A): ");

scanf("%d", &m);

fflush(stdin);

printf("Enter number of columns in the matrix (A): ");

scanf("%d", &n);

fflush(stdin);

printf("\n\n------- MATRIX A -------\n");

// Taking input for matrix A + validation

do {

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

{

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

flag = 1;

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

{

status = scanf("%f", &a[i][j]);

if(status == 0)

{

flag = 0;

break;

}

}

if(flag == 0)

{

printf("Invalid entry!\n");

fflush(stdin);

break;

}

}

}while(flag == 0);

// Transposing the matrices

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

{

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

{

b[j][i] = a[i][j];

}

}

// Printing the matrix C

printf("\n\n------- MATRIX A transpose -------\n");

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

{

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

{

printf(" %0.4f,", b[i][j]);

}

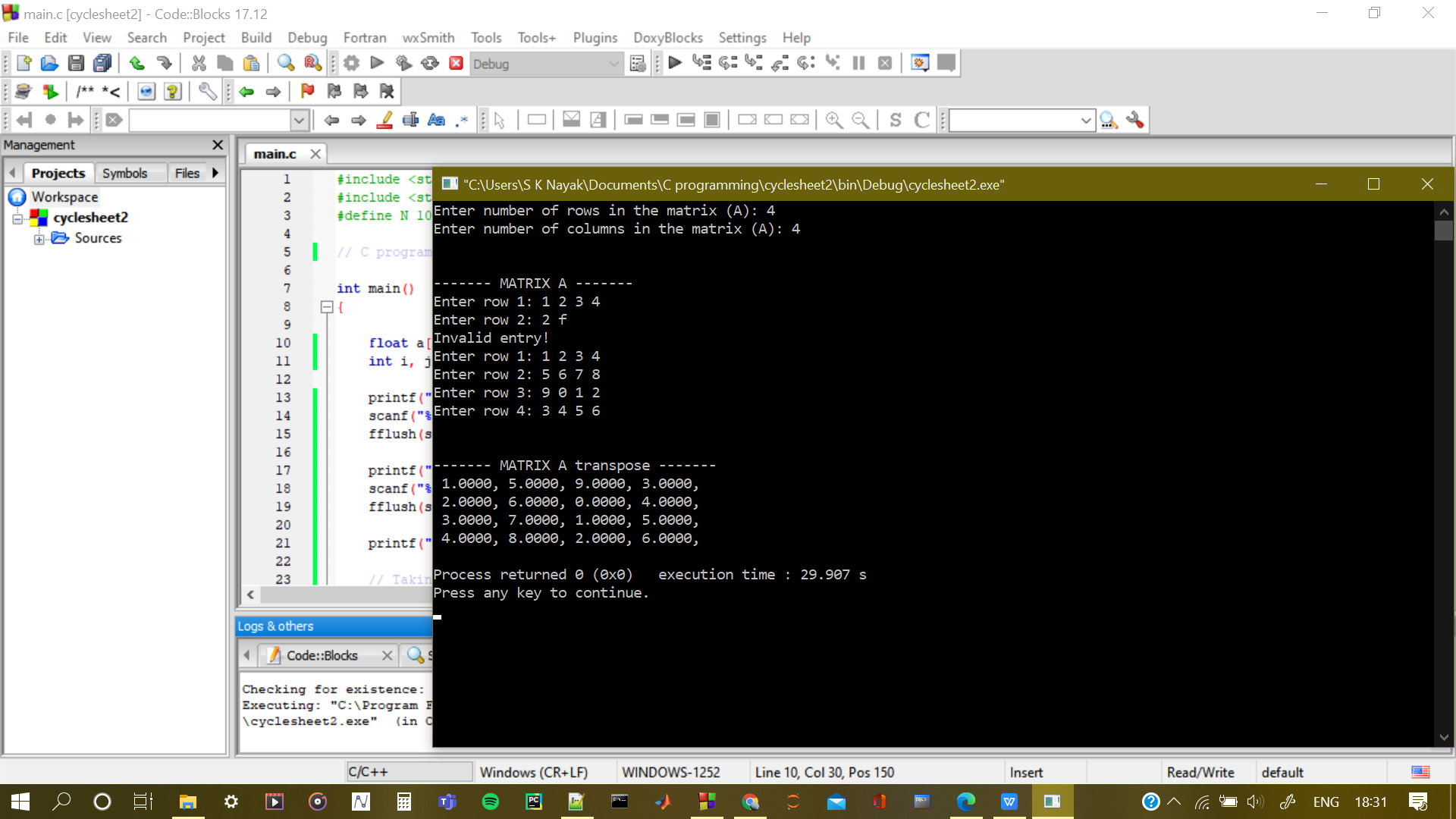
printf("\n");

}

return 0;

}

**Output:**



1. **WAP to the find the product of two matrix**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#define N 50

// C program to multiply two matrices

int main()

{

float a[N][N], b[N][N], c[N][N] = {{0}};// c[N][N];

int i, j, m, n, status, flag, k = 0;

printf("Enter number of rows in the matrix (A): ");

scanf("%d", &m);

fflush(stdin);

printf("Enter number of columns in the matrix (A): ");

scanf("%d", &n);

fflush(stdin);

printf("\n\n------- MATRIX A (m x n) -------\n");

// Taking input for matrix A + validation

do {

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

{

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

flag = 1;

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

{

status = scanf("%f", &a[i][j]);

if(status == 0)

{

flag = 0;

break;

}

}

if(flag == 0)

{

printf("Invalid entry!\n");

fflush(stdin);

break;

}

}

}while(flag == 0);

printf("\n\nThe second matrix has to be %d x %d, otherwise we can't multiply the matrices.\n", n, m);

printf("\n\n------- MATRIX B (n x m) -------\n");

// Taking input for matrix B + validation

do {

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

{

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

flag = 1;

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

{

status = scanf("%f", &b[i][j]);

if(status == 0)

{

flag = 0;

break;

}

}

if(flag == 0)

{

printf("Invalid entry!\n");

fflush(stdin);

break;

}

}

}while(flag == 0);

// Multiplying the matrices

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

{

for(k = 0; k < m; k++)

{

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

{

c[i][k] += a[i][j]\*b[j][k];

}

}

}

// Printing the matrix C

printf("\n\n------- MATRIX A \* MATRIX B -------\n");

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

{

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

{

printf(" %0.2f,", c[i][j]);

}

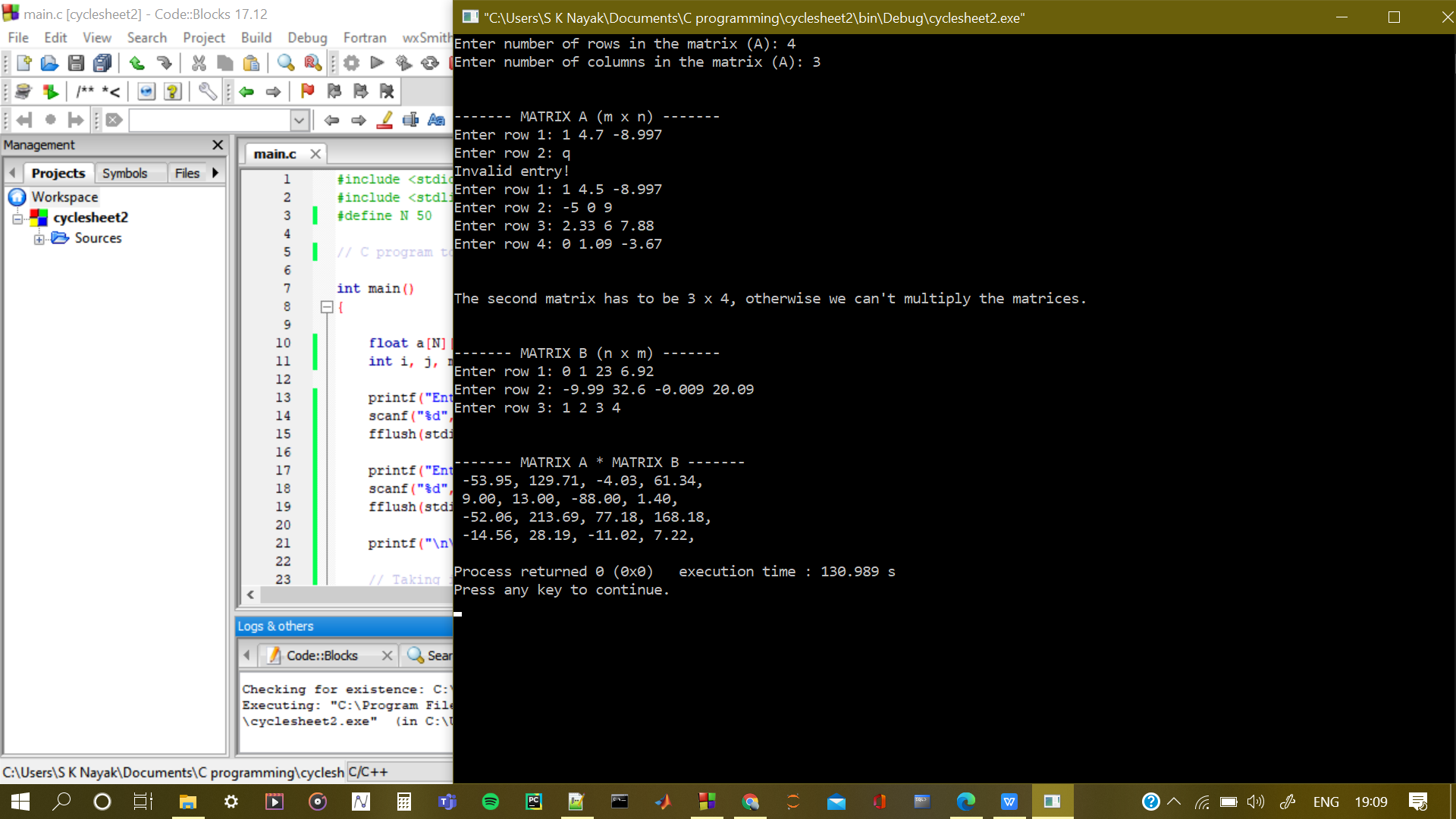
printf("\n");

}

return 0;

}

**Output:**



**CHALLENGING PROGRAMS (ARRAYS)**

1. **WAP to find the product of boundary elements of a given matrix.**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#define N 50

// C program to find sum of boundary elements of a matrix

int main()

{

float a[N][N], border = 0;// c[N][N];

int i, j, m, n, status, flag;

printf("Enter number of rows in the matrix (A): ");

scanf("%d", &m);

fflush(stdin);

printf("Enter number of columns in the matrix (A): ");

scanf("%d", &n);

fflush(stdin);

printf("\n\n------- MATRIX A (m x n) -------\n");

// Taking input for matrix A + validation

do {

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

{

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

flag = 1;

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

{

status = scanf("%f", &a[i][j]);

if(status == 0)

{

flag = 0;

break;

}

}

if(flag == 0)

{

printf("Invalid entry!\n");

fflush(stdin);

break;

}

}

}while(flag == 0);

// Summing up border elements

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

{

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

{

if(i == 0 || i == m-1 || j == 0 || j == n-1)

{

border += a[i][j];

}

}

}

// Printing the matrix A

printf("\n\n------- MATRIX A -------\n");

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

{

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

{

printf(" %0.2f,", a[i][j]);

}

printf("\n");

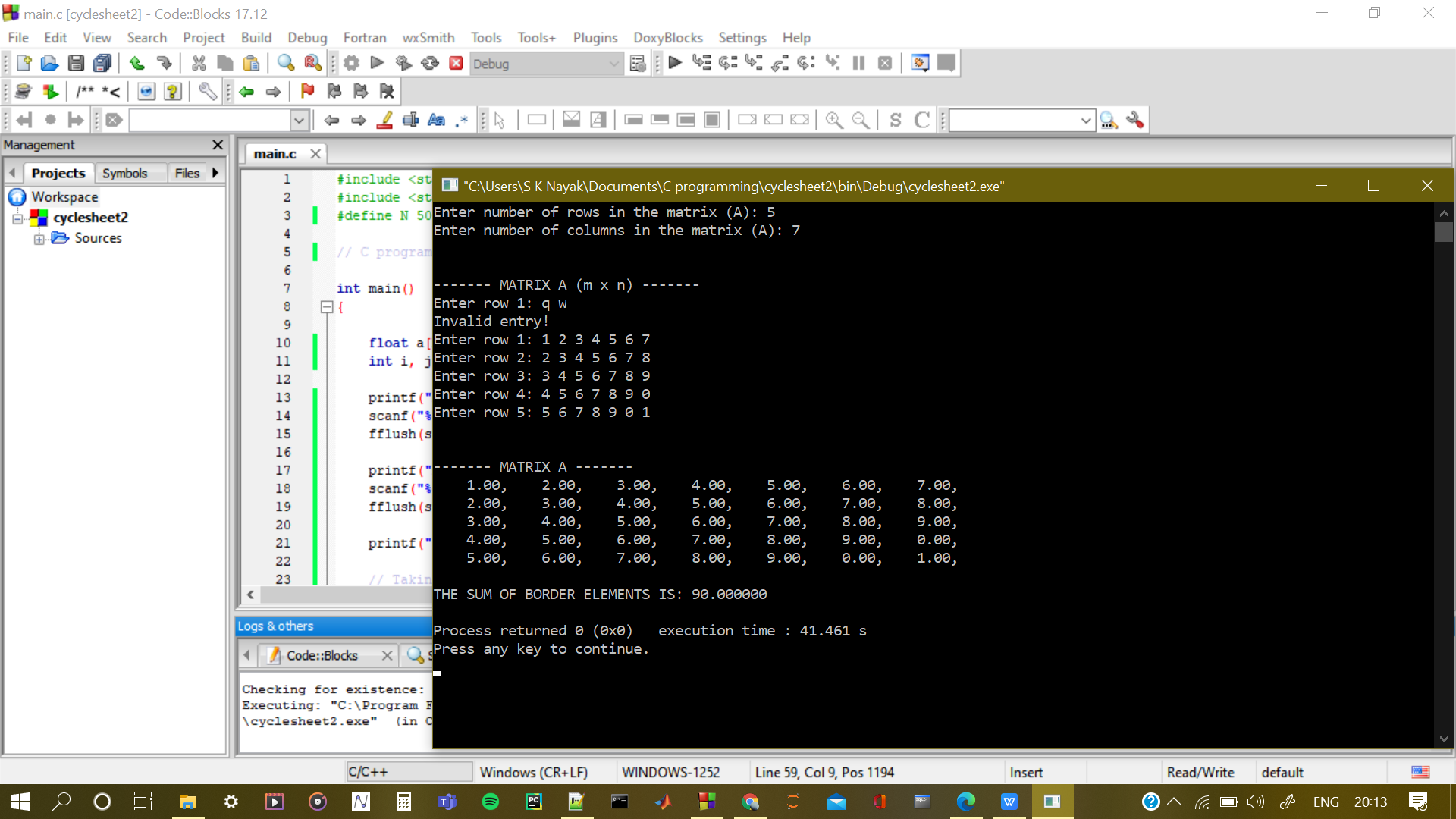
}

printf("\nTHE SUM OF BORDER ELEMENTS IS: %f\n", border);

return 0;

}

**Output:**



1. **WAP to find the maximum element in a given row and minimum element in a given column for a given matrix.**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#define N 50

// C program to find minimum element in a given row

// and maximum element in a given column

int main()

{

float a[N][N], min, max;// c[N][N];

int i, j, m, n, status, flag, pmin, pmax;

printf("\nEnter number of rows in the matrix (A): ");

scanf("%d", &m);

fflush(stdin);

printf("\nEnter number of columns in the matrix (A): ");

scanf("%d", &n);

fflush(stdin);

printf("\n\n------- MATRIX A (m x n) -------\n");

// Taking input for matrix A + validation

do {

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

{

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

flag = 1;

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

{

status = scanf("%f", &a[i][j]);

if(status == 0)

{

flag = 0;

break;

}

}

if(flag == 0)

{

printf("Invalid entry!\n");

fflush(stdin);

break;

}

}

}while(flag == 0);

// Input for row number

fflush(stdin);

printf("\nEnter the row number whose min element you want: ");

status = scanf("%d", &pmin);

// Input validation

while(status == 0 || pmin <= 0 || pmin > m)

{

printf("Invalid row number! Please enter a valid row number: ");

fflush(stdin);

scanf("%d", &pmin);

}

// Input for column number

fflush(stdin);

printf("\nEnter the column number whose max element you want: ");

status = scanf("%d", &pmax);

// Input validation

while(status == 0 || pmax <= 0 || pmax > n)

{

printf("Invalid column number! Please enter a valid column number: ");

fflush(stdin);

scanf("%d", &pmax);

}

// Initial value declaration

min = a[pmin - 1][0];

max = a[0][pmax - 1];

// Finding the min element in the given row

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

{

if(a[pmin - 1][i] <= min)

min = a[pmin - 1][i];

}

// Finding the max element in the given column

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

{

if(a[i][pmax - 1] >= max)

max = a[i][pmax - 1];

}

// Printing the matrix A

printf("\n\n----------- MATRIX A -----------\n");

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

{

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

{

printf(" %0.2f,", a[i][j]);

}

printf("\n");

}

// Printing the min and max elements

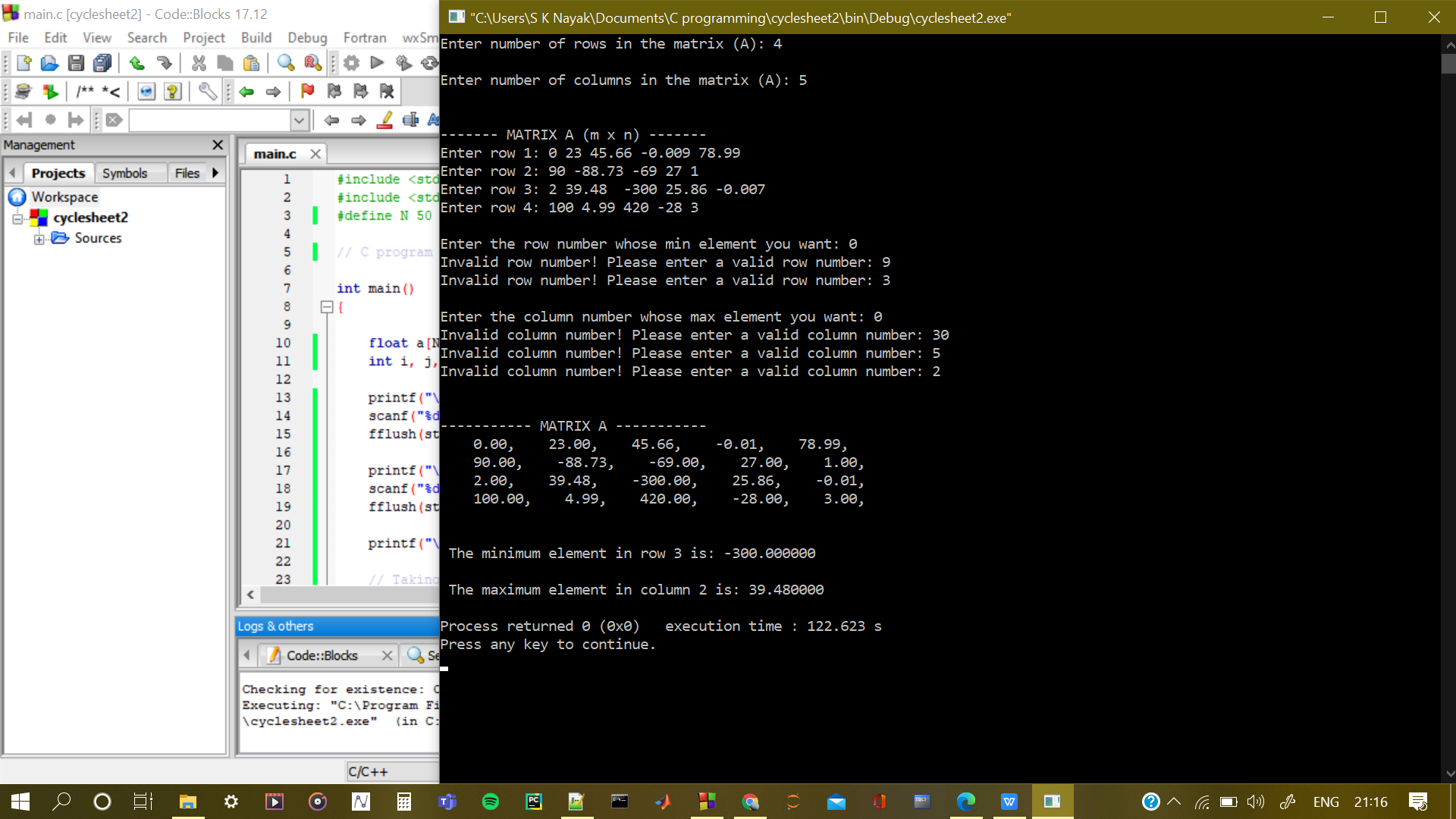
printf("\n\n The minimum element in row %d is: %f", pmin, min);

printf("\n\n The maximum element in column %d is: %f\n", pmax, max);

return 0;

}

**Output:**



1. **Consider the Zig-Zag sequence of the matrix and sum up all the elements in that sequence and replacing non Zig-Zag elements with zero. Finally print the sum of the sequence and the final matrix in matrix format only.**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#define N 50

// C program to find sum of zig-zag elements of a given matrix

// and replace the non zig-zag elements by 0

int main()

{

float a[N][N], sum;// c[N][N];

int i, j, n, status, flag;

printf("\nEnter the dimension of the square matrix (A): ");

scanf("%d", &n);

fflush(stdin);

printf("\n\n------- MATRIX A (m x n) -------\n");

// Taking input for matrix A + validation

do {

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

{

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

flag = 1;

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

{

status = scanf("%f", &a[i][j]);

if(status == 0)

{

flag = 0;

break;

}

}

if(flag == 0)

{

printf("Invalid entry!\n");

fflush(stdin);

break;

}

}

}while(flag == 0);

// Zig-Zaging

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

{

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

{

if(i == 0 || i == 3 || j == n - i - 1)

{

sum += a[i][j];

}

else

{

a[i][j] = 0;

}

}

}

// Printing the matrix A

printf("\n\n----------- MATRIX A -----------\n");

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

{

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

{

printf(" %0.2f,", a[i][j]);

}

printf("\n");

}

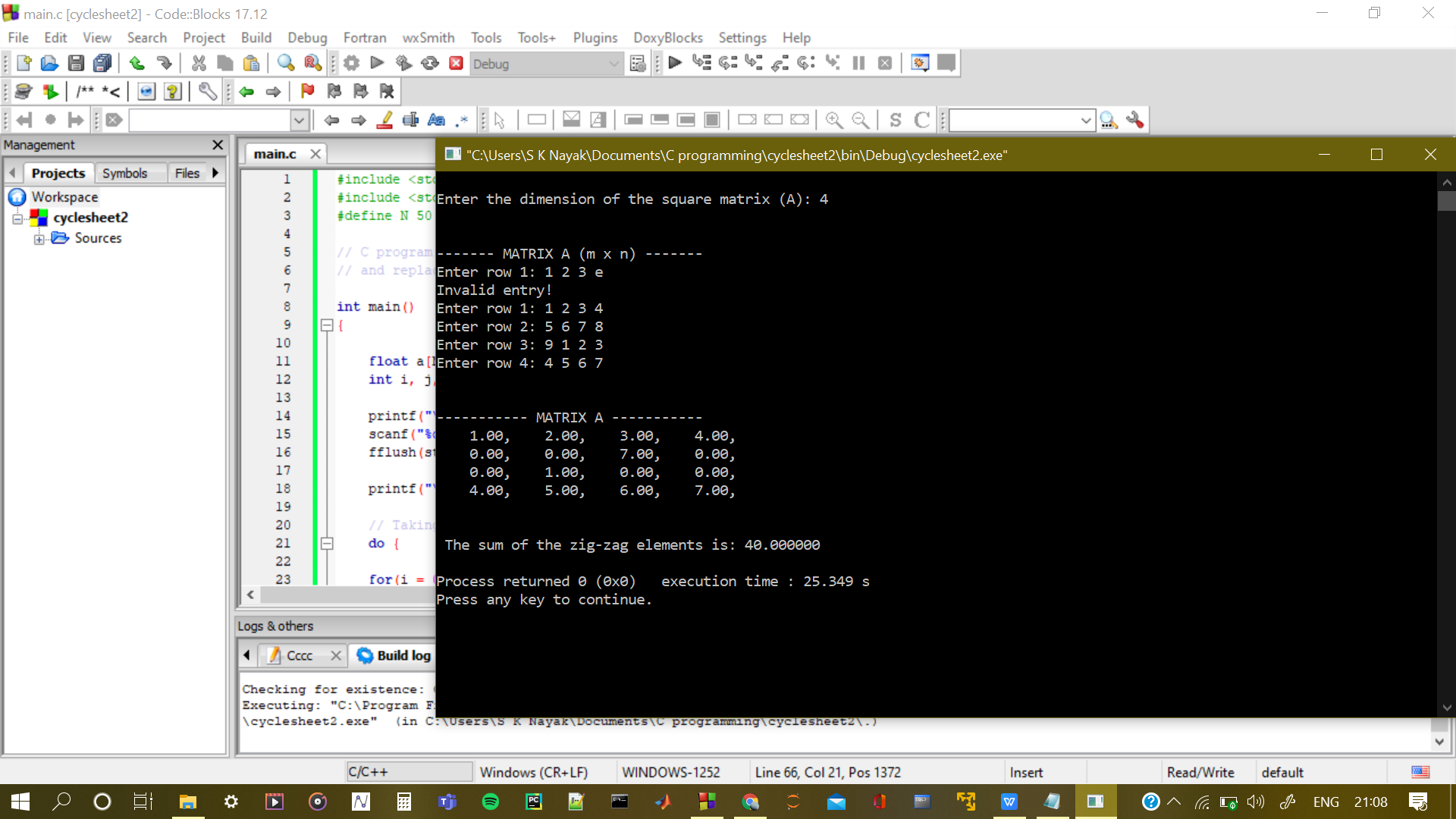
// Printing the min and max elements

printf("\n\n The sum of the zig-zag elements is: %f\n", sum);

return 0;

}

**Output:**



1. **WAP to carry out Left to Right rotations of a 1-D array n times.**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#define N 101

// C program to carry out left to right rotation of an array n times.

void print\_array(char a[N], int l);

int main()

{

char arr[N], temp;

int i, j, n, flag, status;

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

scanf("%d", &n);

fflush(stdin);

// Taking input + validation

do {printf("Enter the array(no spaces unless it's an element): ");

flag = 1;

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

{

status = scanf("%c", &arr[i]);

if(status == 0 || arr[i] == '\n')

{

flag = 0;

break;

}

}

if(flag == 0)

{

printf("Invalid array!\n");

fflush(stdin);

}

}while(flag == 0);

// Right rotating the array and printing it

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

{

temp = arr[0];

for(j = n-1; j > 0; j--)

{

arr[(j+1)%n] = arr[j];

}

arr[1] = temp;

printf("\nRotation %d: ", i+1);

print\_array(arr, n);

}

return 0;

}

void print\_array(char a[N], int l)

{

int i;

printf("{");

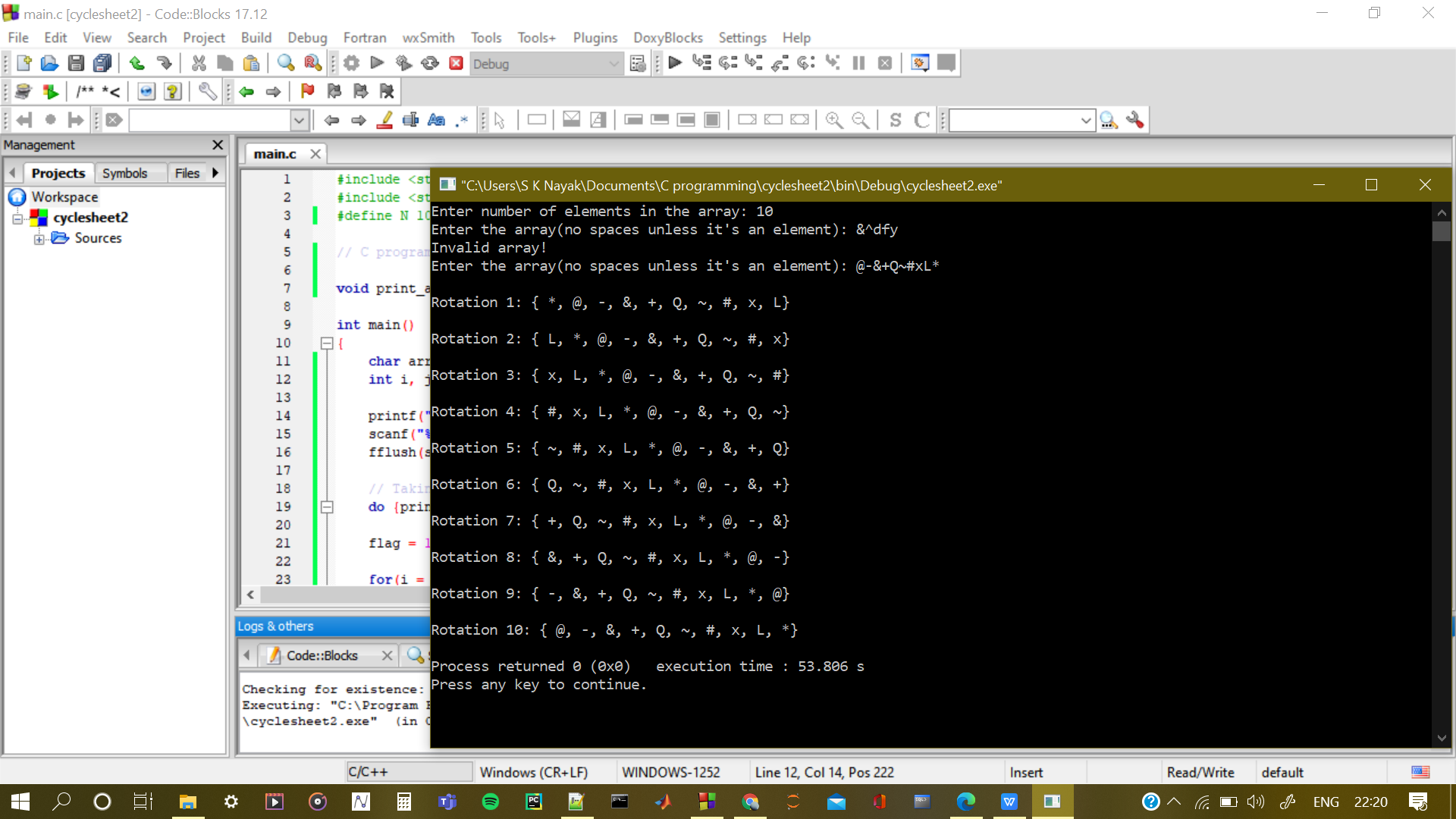
for(i = 0; i < l-1; i++)

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

printf(" %c}\n", a[l-1]);

}

**Output:**



1. **WAP to find all the patterns of 0(2+)0 in a given string. Given a string containing 0’s and 2’s. Find the total number of 0(2+)0 patterns in the string and output it.**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#define N 101

// C program to print out 0(2+)0

int main()

{

int arr[N], n;

int i, j, k, flag, status, exists = 0;

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

scanf("%d", &n);

fflush(stdin);

// Taking input + validation

do {printf("Enter the array(allowed integers are 0,2): ");

flag = 1;

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

{

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

if(status == 0 || arr[i] == '\n' || (arr[i] != 0 && arr[i] != 2))

{

flag = 0;

break;

}

}

if(flag == 0)

{

printf("Invalid array!\n");

fflush(stdin);

}

}while(flag == 0);

// Printing out the sequence

for(i = 0; i < n - 1; i++)

{

// Proceed only if the arr[i] is zero

if(arr[i] == 0)

{

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

{

if(arr[j] == 0)

{

// Checking if all elements in between is 2

for(k = i + 1; k < j; k++)

{

// exists is a flag variable

// if 2 is not present then it will immediately

// break out of the loop

exists = 1;

if(arr[k] != 2)

{

exists = 0;

break;

}

}// third for loop ends

if(j == i+1 && (arr[j] == 0 && arr[i] == 0))

{

exists = 0;

break;

}

// If such a sequence exists then

// break out of loop and go to printing

// j value will be preserved

if(exists == 1)

break;

}//if statement ends

}// Second for loop ends

// Printing the sequence if it exists

if(exists == 1)

{

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

{

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

}

printf("\n");

}

} // If statement ends

}// First for loop ends

return 0;

}

**Output:**