## Assignment: 5

## Exp A. WAP to implement the following scenarios. Take all the input from user, nothing should be imagined or hard coded.

1. Transpose of a matrix.
2. Check if a matrix is Symmetrical or not.

## Coding:

#include <stdio.h>

int main() {

  int a[100][100], transpose[100][100], r, c;

  printf("Enter rows and columns: ");

  scanf("%d %d", &r, &c);

  printf("\nEnter matrix elements:\n");

  for (int i = 0; i < r; ++i)

  for (int j = 0; j < c; ++j) {

    printf("Enter element a%d%d: ", i + 1, j + 1);

    scanf("%d", &a[i][j]);

  }

  printf("\nEntered matrix: \n");

  for (int i = 0; i < r; ++i)

  for (int j = 0; j < c; ++j) {

    printf("%d  ", a[i][j]);

    if (j == c - 1)

    printf("\n");

  }

  for (int i = 0; i < r; ++i)

  for (int j = 0; j < c; ++j) {

    transpose[j][i] = a[i][j];

  }

  printf("\nTranspose of the matrix:\n");

  for (int i = 0; i < c; ++i)

  for (int j = 0; j < r; ++j) {

    printf("%d  ", transpose[i][j]);

    if (j == r - 1)

    printf("\n");

  }

  int Count=1;

  for(int rows = 0; rows < r; rows++)

    {

        for(int columns = 0; columns < c; columns++)

        {

            if(a[rows][columns] != transpose[rows][columns])

            {

                Count++;

                break;

            }

        }

    }

    if(Count == 1)

    {

        printf("\n The Matrix that you entered is a Symmetric Matrix ");

    }

    else

    {

        printf("\n The Matrix that you entered is Not a Symmetric Matrix ");

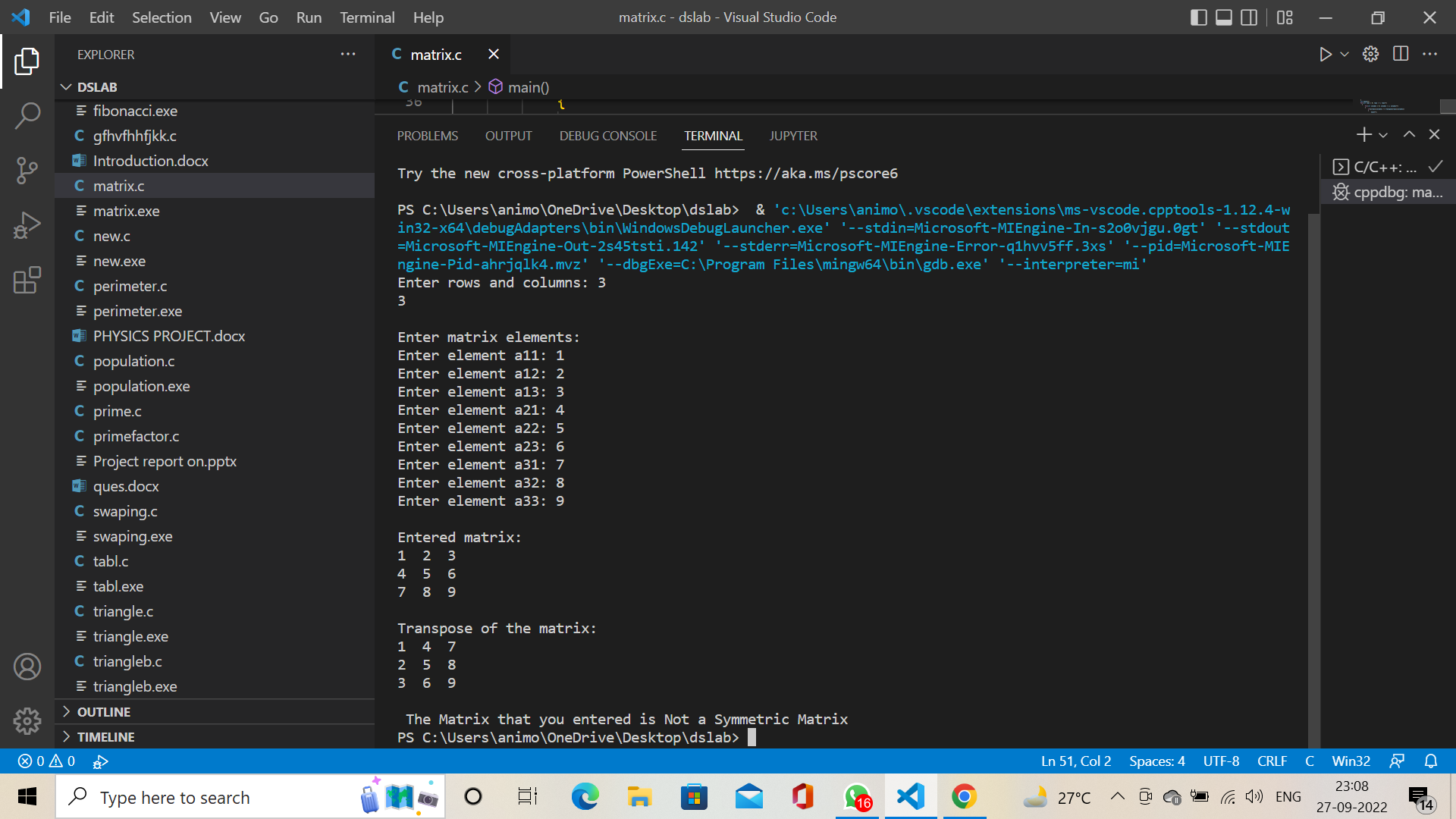
    }

  return 0;

}

## Output:

## 



**Exp B. WAP to merge two arrays and append them in the following order.**

1. Add the first array to the end of another one
2. Add Second Array to the end of the first one
3. Merge the arrays and sort them.

**Coding:**

#include <stdio.h>

int mergearray(int a[], int b[], int arr1size, int arr2size)

{

    int arr\_resultsize = arr1size + arr2size;

    int c[arr\_resultsize], i, j;

    for (i = 0; i < arr1size; i++) {

        c[i] = a[i];

    }

    for (i = 0, j = arr1size;

         j < arr\_resultsize && i < arr2size; i++, j++) {

        c[j] = b[i];

    }

   int aa, n=arr\_resultsize;

   for (i = 0; i < n; ++i){

      for (j = i + 1; j < n; ++j){

         if (c[i] > c[j]){

            aa = c[i];

            c[i] = c[j];

            c[j] = aa;

         }

      }

   }

   printf("The merged sorted array is:\n");

    for (i = 0; i < arr\_resultsize; i++) {

        printf("%d ", c[i]);

    }

}

int main()

{

    int arr1size = 5, arr2size = 5;

    int a[5] = { 1, 2, 4, 4, 8 };

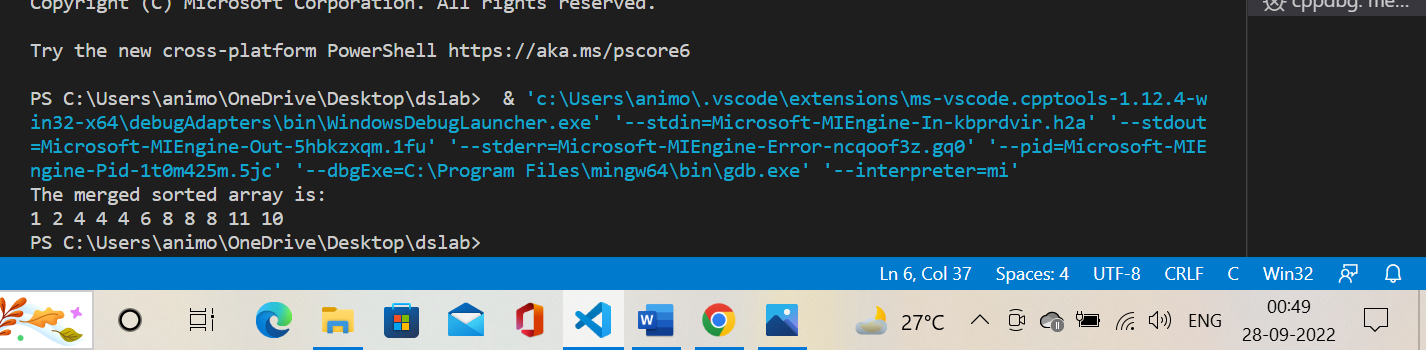
    int b[5] = { 6, 8, 8, 11, 4 };

    printf("%d", mergearray(a, b, arr1size, arr2size));

    return 0;

}

**Output:**



**Exp C.** WAP using pointers to find the smallest number in an array using pointer.

**Coding:**

# include <stdio.h>

int  main()

{

int  arr[50],n,i,small ;

int  \*ptr ;

printf("Enter the length of numbers: ");

scanf("%d", &n) ;

printf("Please enter %d number : \n",n);

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

{

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

ptr++ ;

}

ptr = & arr[0];

small = arr[0];

{

if ( small > ( \*ptr ))

small = \*ptr ;

ptr++ ;

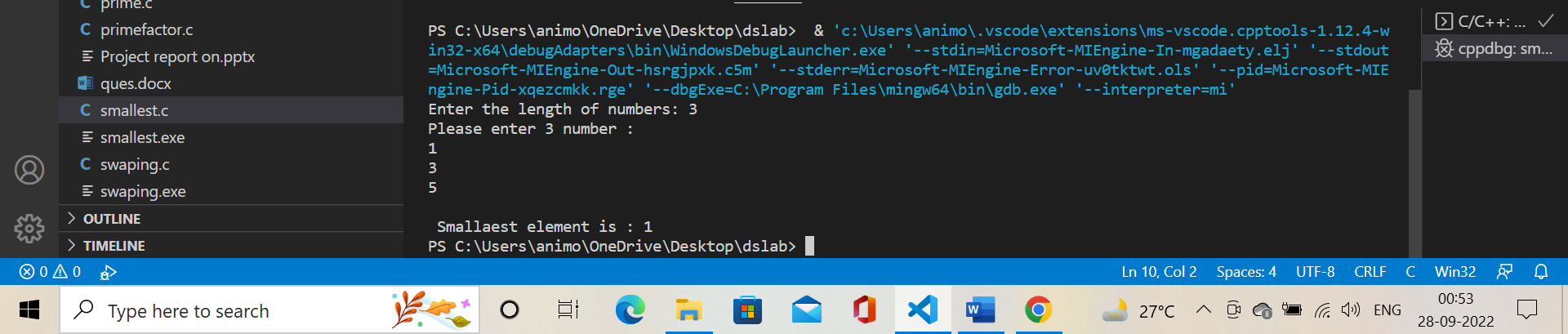
}

printf("\n Smallaest element is : %d",small) ;

return 0;

}

**Output:**



**Exp D.** WAP which performs following task.

1. Initialize an integer array of 10 elements in main()
2. Pass the entire array to a function modify()
3. In modify() multiply(you can use division, addition or subtraction) each element of array by 3
4. Return the control to main() and print the new array elements in main().

**Coding:**

#include<stdio.h>

void modify(int data[]){

    for(int i = 0; i < 10; i++) data[i] \*= 3;

}

void main()

{

    int i;

    int data[] = {1,2,3,4,5,6,7,8,9,10};

    modify(data);

    printf("Multiplied array : ");

    for(i=0;i<10;i++){

        printf("%d ",data[i]);

    }

}

**Output:**

