**Foundation of algorithms**

**IE2072**

A blue and orange logo

Description automatically generated with medium confidence

IT21831904

Weerasinghe K.M

Contents

[Question 01 3](#_Toc148737710)

[Output 5](#_Toc148737711)

[Question 02 6](#_Toc148737712)

[Output 8](#_Toc148737713)

# Question 01

#include <stdio.h>

int main() {

    int N, M; // Declare N and M.

    int k;

    // Prompt the user to input the matrix size

    printf("input the matrix size:\n");

    scanf("%d %d", &N, &M);

    // Declare a matrix array with N rows and M columns.

    int matrix[N][M];

    // user to input matrix elements.

    printf("input Matrix elements:\n");

    int i, j; // Declare loop counters .

    for (i = 0; i < N; i++) {

        for (j = 0; j < M; j++) {

            scanf("%d", &matrix[i][j]); //  matrix elements in the array.

        }

    }

    int X = 0, Row\_Val, Col\_Val; // Declare variables for X, Row\_Val, and Col\_Val.

    // Iterate while Row\_Val or Col\_Val is not equal to 1, or until a certain condition is met.

    for (k = 0; k < (N \* M); k++) {

        for (i = 0; i < N; i++) {

            Row\_Val = 1; // Initialize Row\_Val to 1.

            for (j = 0; j < M; j++) {

                // Check if any element in the current row is less than X.

                if (X > matrix[i][j]) {

                    Row\_Val = 0; // If yes, set Row\_Val to 0 and break out of the loop.

                    break;

                }

            }

            if (j == M) // If j reaches M, it means all elements in the row were greater or equal to X.

                break;

        }

        for (i = 0; i < M; i++) {

            Col\_Val = 1; // Initialize Col\_Val to 1.

            for (j = 0; j < N; j++) {

                // Check if any element in the current column is less than X.

                if (X > matrix[j][i]) {

                    Col\_Val = 0; // If yes, set Col\_Val to 0 and break out of the loop.

                    break;

                }

            }

            if (j == N) // If j reaches N, it means all elements in the column were greater or equal to X.

                break;

        }

        //  increment X.

        if (Col\_Val == 1 && Row\_Val == 1)

            X++;

        // no, break the loop.

        else

            break;

    }

    int result = X - 1; // Calculate the result final X value.

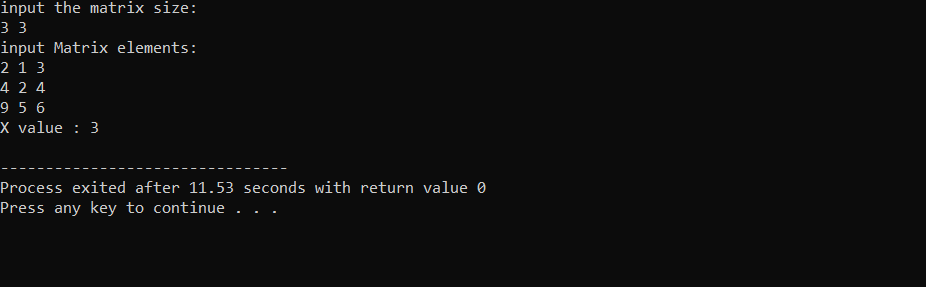
    // Print the result

    printf("X value: %d\n", result);

    return 0;

}

# Output



# Question 02

#include <stdio.h>

int main() {

    int N, K, count = 0,i;

    // User input for the size of the array frequency

    printf("Enter the size of the array frequency: ");

    scanf("%d", &N);

    int frequency[N]; // Declare the frequency array

    // User input for elements of the array frequency

    printf("Enter elements of the array frequency: ");

    for (i = 0; i < N; i++) {

        scanf("%d", &frequency[i]);

    }

    // User input for the size of the array Key\_size

    printf("Enter the size of the array Key\_size: ");

    scanf("%d", &K);

    int key\_size[K], num\_keys = 0; // Declare the key\_size array

    // User input for elements of the array Key\_size

    printf("Enter elements of the array Key\_size: ");

    for (i = 0; i < K; i++) {

        scanf("%d", &key\_size[i]);

        num\_keys += key\_size[i]; // Calculate total keys

    }

    if (num\_keys < N) {

        printf("-1\n"); // Not enough keys for the frequency array

    } else {

        // Sort the frequency array in descending order

        for (i = 1; i < N; i++) {

            int current = frequency[i];

            int j = i - 1;

            while (j >= 0 && frequency[j] < current) {

                frequency[j + 1] = frequency[j];

                j--;

            }

            frequency[j + 1] = current;

        }

        printf("The array in descending order is:\n");

        for (i = 0; i < N; i++) {

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

        }

        int loop\_count = 0;

        int j = 1;

        while (loop\_count < N) {

            int i = 0;

            int available\_keys = 0;

            // Count how many keys are available in the current loop

            for (i = 0; i < K && loop\_count < N; i++) {

                if (key\_size[i] > 0) {

                    available\_keys++;

                    loop\_count++;

                }

            }

            // Distribute available keys and calculate the cost

            for (i = 0; i < available\_keys; i++) {

                key\_size[i]--;

                count += (frequency[loop\_count - available\_keys + i] \* j);

            }

            j++; // Increment j for the next iteration

        }

        printf("\nThe minimum cost is %d\n", count);

    }

    return 0;

}

# Output

