In this task there are 3 exercises ( the exercise should only .c .h and mikefile files)

The program will show a menu that allows the user to select the exercise they were to run, or another option to exit. The main menu will be presented as long as the user did not select to exit.

**To present the menu need to use switch structure, need to make sure the user can type capital or lower-case letters.**

For every option on the menu need to write in a different function there should be no code in the switch except to call for the functions.

1. All functions that get matrix as an input, will get it as the address of the beginning of the memory of the matrix (one asterisk(.
2. The matrixes will be defined only in the main function of every exercise, and their size will be set as a constant variable at define
3. Although Ubuntu allows it- do not define a matrix with a parameter as in the example! Only by a constant:

int x; #define N 8

int arr[x]; int mat[N]

Example of output is at the end

**Exercise one: multiplication of matrixes:**

In this excursive we will be multiplicate two square matrixes from int type A and B with size N\*N to a third matrix C according to the matrix multiplication rules.

1. In the first function of the exercise doMultMat define 3 square matrixes, with size N\*N, choose a value as you want, print this value to the screen.
2. Get input from the user of a number that will define the actual size of the matrix A so that the number of rows and columns will be between 1 and N included. Print this number to the screen
3. Set matrixes A and B with random numbers, choose a range as you want, set as constants. Print the values to the screen
4. Print matrixes A and B
5. Execute and run the multiplication function multMatrix that get three matrixes as an input A, B and C and additional required data, and return the multifaction result in matrix C.
6. Print matrix C

**Exercise 2: Adding matrixes**

1. In this exercise we will add two matrixes A and B with size M\*N to a third matrix C according to the sum of Matrixes rules.
2. In the main function of the exercise doSumMat, define 3 matrixes with size ‘ROWS’ rows and ‘COLS’ columns, select values as you want. Print these values to the screen.
3. Get as an input two numbers that will define the actual size of matrix A, so that the number of rows is between 1 and ROWS, and the number of columns is between 1 and COLS including. Print these values to the screen.
4. According to the sum of matrixes rules, the size of matrix C is determined according to section 2.
5. Execute and run the sum function sumMatrix that gets three matrixes an an input A, B, and C, and additional required information, and returns the result of Sum of A and B at matrix C
6. Print matrix C

**Note: Note that there are identical processes in Exercise 1 and 2, use functions wisely**

**Exercise 3: Buying Festival Tickets**

In this exercise we will run a system of buying tickets for a festival that lasts X days. Execute a main function for the doTickets exercise.

Work process (must be divided into functions)

1. Capture the days on which the festival takes place 1 marks Sunday 7 Saturday.

2. Ask the user the day he wants to arrive and how many tickets he wants to buy for that day. If the user entered 0, we would know that he no longer wants to buy tickets

3. After the sale of all tickets, print in the main function of the exercise

What is the day to which the most tickets were booked, and how many were they.

What is the day to which the least tickets were booked, and how many were they.

The work should be divided into auxiliary (help) functions.

The following functions must be used:

1. getDayAndTickets - returns 0 if the user did not want to continue buying tickets and 1 if he bought tickets. In addition, the function returns by using pointers the day of the bookings and how many tickets were bought for that day. The function can have more variables according to your decision

FindMaxMinArr - which receives an array and its length and other data according to your decision. The function returns using pointers the day on which the maximum number of tickets was purchased and the day on which the minimum number of tickets was purchased.

Note: All input numbers must be input tested. It can be assumed that if they asked for a number the user entered a number.

Hint - You can set up a size 7 array and use only part of it

A picture containing text

Description automatically generatedText

Description automatically generated