Section 16

PDS Lab **Lab-7** **13.10.2023**

Instructions:

- This lab is based on the topics: String, 2D Array
- You should save each program with the file name as specified against each problem as <Lab#>_<Assignment#>_<Roll#>.c. For example, **07_01_23CS10006.c** to save Program to 1st assignment in Lab 7 with Roll Number 23CS10006
- You should upload each program to the Moodle system. Also, copy+paste your programs to the text window on the test page.
- There will be no evaluation and hence grade, if you don't submit your .c files to the Moodle server. Use **emacs** editor and **gcc command** in terminal to run the following programs.
- Document your programs meaningfully using appropriately named variable and sufficient amount of comments. Documentation and proper code indentation carry marks.
- Do not use advanced concepts like structures anywhere in your code.
- **The top two lines of your programs must contain the following information:**

```
//Roll No.: <Type in your roll no.>
//Name: <Type in your name>
```

SET A

1. Write a C program having the following functions.
   main: Define a 20 by 20 matrix named as **data.** In the main function first read two integers m and n (2<m<20 and 2<n<20) and fill an **m×n** submatrix of the matrix **data** with random integers in the range [0, 10] by calling the **rand** function. Display the matrix **data** nicely formatted. Then call three functions **findMax**, **findMin, findMinMax** .

   findMax: This function should take matrix **data**, the dimensions of the submatrix **m** and **n** (also any other parameters that you may need) as its arguments and find a 3×3 submatrix for which sum of all elements is maximum. It should display this submatrix nicely formatted. If no unique submatrix having maximum sum is present, you can display any one of the submatrix having maximum sum.

   findMin: This function should take matrix **data**, the dimensions of the submatrix **m** and **n** (also any other parameters that you may need) as its arguments and find a 3×3 submatrix whose sum of all elements is minimum. It should display this submatrix nicely formatted along with the sum of all its element. If no unique submatrix having minimum sum is present, you can display any one of the submatrix having minimum sum.

   Example: Consider m=4 and n=4, and the following example random array:

```
6  0  5  3
2  7  3  9
1  5  7  8
3  4  6  7
```

```
Max submatrix:  7  3  9

                5  7  8

                4  6  7

Sum=56

Min submatrix:   6  0  5

                 2  7  3

               1  5  7
Sum=36
```

findMinMax: This function should take matrix **data**, the dimensions of the submatrix **m** and **n** (also any other parameters that you may need) as its arguments. It should first find the maximum entry of each of the rows and then find the minimum of all those maximum entries to fin the **minMax**. Display the minMax along with the row and column number in which it is present. If no unique **minMax** is present, you can display any one.

Example: Consider m=4 and n=4, and the following example random array:

```
6  0  5  3
2  7  3  9
2  5  7  8
4  4  6  7
```

Max for the rows is [6,9,8,7]. Minimum of these is 6.

The display should be: "minMax=6, present in row 0, column 0".

findMaxMin: This function should take matrix **data**, the dimensions of the submatrix **m** and **n** (also any other parameters that you may need) as its arguments. It should first find the minimum entry of each of the rows and then find the maximum of all those minimum entries to fin the **maxMin**. Display the maxMin along with the row and column number in which it is present. If no unique **maxMin** is present, you can display any one.

Example: Consider m=4 and n=4, and the following example random array:

```
6  0  5  3
2  7  3  9
3  5  7  8
5  4  6  7
```

Min for the rows is [0,2,3,4]. Minimum of these is 4.

The display should be: "maxMin=4, present in row 3, column 1".

2. In the main program, read a character string **str** of size at most 20 from the user and display it. Then call the functions **check**, **encode1** and **encode2**. Finally, the string **str** should again be displayed.

   a) **check:** This function should take a string of size of at most 20 as its parameter and check how many times the letter 'a' to 'y' appear before the succeeding character (i.e. 'b' to 'z') separated by exactly 1 character in the string and display the result.

      Example 1. Input string "acdb": Expected output: 0

      Example 2. Input string "acbdbe": Expected output: 3

      Example 3. Input string "aaabbbb": Expected output: 2

   b) **encode1:** This function should take a character string of size at most 20 as parameter and then encode the characters of the string as follows: a → c, b→ d,…, y→a,z→b. The string received as argument should be undisturbed. It should then display the encoded string.

      Example 1. Input string "acdb": Expected output: cefd

      Example 2. Input string "acbdbe": Expected output: cedfdg

      Example 3. Input string "aaabxyz": Expected output: cccdzab

   c) **encode2:** This function should take a character string of size at most 20 as parameter and substitute the character 'a' before every character in the string. The string received as argument should be undisturbed. It should then display the encoded string.

      Example 1. Input string "a": Expected output: "aa"
      Example 2. Input string "acdb": "aaacadab"
      Example 3. Input string "aaab": Expected output: "aaaaaaab"

3. Write a C program with the following functions. First, declare a global **5×5** integer array named **dist**.

   a) **main:** Fill the array **dist** with random values in the range [20, 999] such that the upper triangle and lower triangle are symmetrical. The principal diagonal should be filled with 0s. Assume that the array **dist** represents the direct distance between 5 cities (A,B, C, D, and E) in Km. Display the generated **dist** array with proper formatting. Now make all distances more than 150km as -1 indicating disconnected cities. Again display the **dist** array with proper formatting. Next, call the following two functions **dist2city, distThr1city**.

   b) **dist2city**: Read from keyboard two characters in the range A to E (say **s** and **t**). Determine the direct distance between the cities **s** to **t** and display it.

   d) **DistThr1city**: Read from keyboard two characters in the range A to E (say **s** and **r, s** ≠ **r**), find the distance between the cities **s** and **r** through one intermediate city and display these. Display the minimum distance between cities s and r through an intermediate city.

   Example Display:

```
From A to C via B unreachable

From A to C via D  200km

From A to C via E   250 km   etc.

Minimum distance from A to C is via D and is 200Km
```

SET B

1. Write a C program to fill a single dimensional integer array of size 50 with random integral numbers in the range [10, 50] by appropriately calling the rand() library function. User will first enter the number of such numbers to be generated (maximum 50).

   Display the array contents. Then, determine all the triplets that add up to 60 and display the corresponding array indices and the values stored in those locations.

2. Write a C program to read the roll number (integer), age (integer) and marks of 20 students admitted to a department. Generate random roll numbers in the range [1000,2000], age in the range [15 to 25] and marks in the range [0,100] and populate the respective arrays.. Please use three one dimensional arrays for storing roll number, age, and mark.

    a. Display the roll number, ages, and marks of all students having the same age.
    Example display:
    ```
    Roll: 1025   Age: 20   Mark:83
    Roll: 3021   Age:20    Mark:45
          . . . . . . . . . . . .  . . .
    Roll   2450    Age:21    Mark:47
    Roll   1975   Age 21:   Mark:59
    ```

    b. Display the roll number, ages, and marks of all students having identical marks.
    Example display:
    ```
    Roll: 1027   Age: 23   Mark:85
    Roll: 3025   Age:25    Mark:85
    . . . . . . .    . .. . . . . . . . . . . . . . .
    Roll   2459    Age:21    Mark:77
    Roll   1990   Age:23   Mark:77
    ```

    c. Sort the students according to their roll numbers and display the details nicely formatted.
    ```
      Roll: 1027   Age: 23   Mark:85
      Roll   1990   Age:23   Mark:77
    . . . . . . .    . .. . . . . . . . . . . . . . .
      Roll   2459  Age:21    Mark:77
      Roll: 3025   Age:25    Mark:85
    ```

3. Write a C program to read from keyboard the roll number (unsigned integer) and CGPA (float) of up to 10 students. User will first enter the number of such students for which inputs will have to be taken (maximum 10). Use two arrays of maximum size of 10 each.

   a. Display the details read, nicely formatted
   b. Display the average CGPA of the students.
   c. Display the roll numbers and CGPA of all students having identical CGPA (for checking equality of CGPA, consider two digit accuracy).