National University of Computer & Emerging Sciences, Karachi
CS-Department
FAST School of Computing
Lab Final Examination, Fall 2023
11th Dec, 2023, 08:30 am - 10:30 am

Course Code: CL1002	**Course Name: Programming Fundamentals Lab**
Instructor Names: Fahad Hussain, Sir Taha Ahmed, Miss Zainab Asif	
Student Roll No:	**Section No:**

General Instructions:

- Carefully read the following instructions before attempting the paper.
- Except your Roll No and Section, DO NOT WRITE anything on this paper.
- The Exam consists of 3 questions on 2 printed sides of 1 page.
- In case of any ambiguity, you may make assumptions, but your assumption must not contradict any question.

Submission Instructions:

- You must comment your student ID on top of each file.
- Name the .c file for each question according to Roll_No e.g. k23xxxxQ1.c, k23xxxxQ2.c etc.
- Create a folder with the name K23XXXX accroding to your student id.
- Put all your C files (1 for each question) in the above folder. Then paste this folder on the local share.
- Submissions are on local storage that can be accessed using win+R keys and entering \\172.16.5.43

Total Time: 120 minutes ***Maximum Points: 50***

1. File Processing

[LLO: 4, Marks=15, Time 35min]

Develop a C program that manages product and category information through file operations. The program offers the following functionalities:

Add Categories: Users can add new categories to categories.txt, providing a descriptive category name followed by a unique ID per line (e.g., Fruits 1).

Add Products: Users can append product data to products.txt. Each line in products.txt should contain a descriptive product name, its price, and the associated category ID (e.g., Apple 2.50 1).

Generate Final Result: Upon execution, the program reads data from products.txt and categories.txt. It organizes products by category and creates an output file named final.txt. Each line in final.txt displays the descriptive category names along with corresponding product names and prices (e.g., Fruits Apple 2.50).

The program showcases a menu allowing users to add categories, add products, or generate the final.txt file. It continues to display the menu options until the user chooses to exit by entering '0'. This menu-driven approach ensures the user can select options (1, 2, 3, or 0) to perform desired operations in the program.

Categories.txt

```
Fruits 1
Electronics 2
```

products.txt

```
Apple 2.50 1
TV 500.00 2
Orange 1.80 1
Laptop 1200.00 2
Banana 1.20 1
```

final.txt

```
Fruits Orange 1.80
Fruits Apple 2.50
Fruits Banana 1.20
Electronics TV 500.00
Electronics Laptop 1200.00
```

2. Pointers [LLO: 03, Marks=15, Time 40min]

Generate a 2D array with dimensions MxN with dynamically allocated memory. User keeps entering rows of matrix until he/she types a '-1'. Number columns will be given in first line.
Your task is to allocate a new 2D array of dimensions M-2 x N-2, where result[i][j] = sum of the 3x3 matrix centered at i+1, j+1.

Example Input:

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

Example Output:

Resultant matrix dimension = 2x4

```
30 39
42 51
30 39
48 49
```

Explanation:
Let's take result[0][0] as an example. This is the sum of the 3x3 matrix centered at matrix[1][1]:

```
1 2 3
5 6 7
1 2 3
```

By summing the above elements, we get 30. Same is the process for other cells of the resultant array.

3. Structures and Functions [LLO: 2, Marks=20, Time 45min]

Imagine you are developing a simple text-based RPG game, and you want to represent different types of game characters using a structure. Each character has specific attributes such as name, health points (HP), attack points (AP), and defense points (DP). To manage this information efficiently, you decide to use a structure named **GameCharacter**. But be considerate of the following restrictions: -

a. The sum HP+AP+DP is exactly equal to 10.
b. HP, AP and DP all are less than or equal to 5 and greater than 0.
c. No two points can be same.
d. The name length is no more than 80 chars long.

Write a C program that defines the **GameCharacter** structure and allows users to interact with game characters.

Implement the following by creating these functions:

1. **Create Characters**: Allow the user to input details for multiple game characters until they choose to stop with esc key.
2. **Display Characters**: Print the details of all the game characters entered, including the total count.
3. **Battle Simulation**: Simulate a battle scenario where two characters fight. Allow the user to choose two characters for the battle, deduct health points based on attack and defense points, and declare the winner based on remaining health points.