**Home Assignment**

**Exercise 1**

INPUT → String →**Sample input**

**"Barca 1 Sevilla 0, Real Madrid 1 Barca 1, Barca 3 Valencia 1, Villarreal 2 Barca 1, Espanyol 2 Barca 4";**

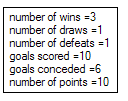
INPUT → TEAM NAME → **Barca**

Write a program to work out how many wins Barca had, how many games they drew, and how many Barca lost. & how many games are playing at home.

Extend the program to work out how many goals Barca scored and how many they conceded.

Suppose a win gains you 3 points, a draw 1 point, and a loss no points. Have your program work out how many points in total Barca have acquired.

When you have finished this exercise, the **output should be as follows:**



**Exercise 2**

Write a program that take an integer argument n and prints all n! permutations of the n letters starting at a (assume that n is no greater than 26). A permutation of n elements is one of the n! possible orderings of the elements.

As an example, when n = 3 you should get the following output (but do not worry about the order in which you enumerate them):

**bca cba cab acb bac abc**

**Exercise 3**

You are working as a construction project manager responsible for allocating tasks to your workers for a large- scale construction project. Each worker is assigned multiple tasks, such as excavation, painting, masonry, or plumbing. Tasks are categorized into levels based on the time required for completion:

1. Level 01: Tasks that take more than 8 hours (e.g., concrete pouring, heavy machinery operation).
2. Level 02: Tasks that can be completed in 7–8 hours (e.g., painting a single room, laying tiles).
3. Level 03: Tasks that take 2–3 hours (e.g., assembling small structures, drilling, or inspections).

Your task is to implement a program to manage the workers and their assigned tasks, log their work hours, and generate reports.

**Requirements**

1. Struct Definitions

Create two structs to represent workers and tasks:

**struct Task {**

**int taskID; // Unique ID for the task int level; // Task level (1, 2, or 3)**

**int estTime; // Estimated time for the task int loggedTime; // Time logged for the task**

**};**

**struct Worker {**

**int workerID; // Unique ID for the worker**

**int totalLoggedTime; // Total time logged by the worker**

**Task tasks[3]; // Array to store up to 3 tasks for the worker int taskCount; // Number of tasks currently assigned**

**};**

Use these structs throughout the program.

1. Assign a Task to a Worker

Write a function to assign a task to a specific worker:

# void AssignTask(struct Worker\* workerPtr, int tskID);

* + The function should assign the task with the given ID to the worker.
  + Ensure that no more than three tasks are assigned to a worker.

1. Assign Tasks to Multiple Workers

Implement a function to distribute tasks among multiple workers:

# void AssignBulTasks(int ws, struct Worker workerList[ws], int ts, struct Task taskList[ts]);

* + This function assigns tasks from a given list to multiple workers.
  + If there are leftover tasks after assignment, they should be clearly mentioned in the output.
  + Each worker can only have three tasks at most.

1. Log Time for a Task

Write a function to log time spent on a task:

# void LogTaskTime(struct Worker\* workerPtr, struct Task\* tskPtr, int time);

* + Update the task’s logged time.
  + Add the time to the worker’s total logged time.

1. Print Task Time Log

Implement a function to display the details of a task:

# void PrintTaskTimeLog(struct Task tsk);

* + Show the task level (Level 01, 02, or 03).
  + Show the estimated time and the logged time for the task.

1. Print Assigned Tasks for a Worker

Create a function to print all tasks assigned to a worker:

# void PrintWorkerTaskList(struct Worker worker);

* + Display the task IDs, levels, and estimated times of the tasks assigned to the worker.

1. Print Worker Time Details

Write a function to generate a summary for a worker:

# void PrintWorkerTimeDetails(struct Worker worker);

* + Include the total time assigned, total time logged, and any remaining or excess time.

1. Save Worker Data to File

Implement a function to save details of multiple workers to a file

# void PrintWorkersInTxtFile(int ws, struct Worker workerList[ws], FILE \*fptr);

* + Save the details of each worker and their assigned tasks in a text file.

1. Create New Worker

Write a function to create a new worker:

**struct Worker NewWorker();**

* + Initialize a worker with default values.

1. Create New Task

Write a function to create a new task:

# struct Task NewTask();

* + Initialize a task with default values.

**Submission Guidelines**

1. The program must handle multiple workers (use an array of struct Worker).
2. Include at least 4 workers and 10 tasks in your implementation.
3. Add clear comments explaining the logic for each function.
4. Ensure proper memory management and error handling (e.g., avoid assigning more than 3 tasks to a worker).

**Sample Output**

Input:

Enter the number of workers: 3 Enter the number of tasks: 5 Assigning tasks...

Output:

Worker 1:

* Task 1: Level 01, Est. Time: 9 hours
* Task 2: Level 03, Est. Time: 3 hours

Worker 2:

* Task 3: Level 02, Est. Time: 7 hours
* Task 4: Level 03, Est. Time: 2 hours

Leftover Task:

* Task 5: Level 03, Est. Time: 3 hours Log Time:

Enter Worker ID: 1 Enter Task ID: 1 Enter Time Logged: 5

Task 1 Updated:

* Logged Time: 5 hours
* Remaining Time: 4 hours