Applied C Programing Problem Statement

LTTS GLOBAL ENGINEERING ACADEMY

L&T Technology Services

1. Problem Description:

A Sales manager of a company wants to make sales analysis at the end of the financial year.
For that he is collecting the data from shops located in different locations in the below format as a comma separated value(CSV) file.
The file contains data as shown below (actual file may contain more lines):

"sales.csv"

```
Bengaluru, Bajaj, 934563, 120.95, Jan
Delhi, Philips, 1287, 160.69, Feb
Hyderabad, HP, 65576, 350.00, Mar
…
…
```

The data is a sequence of collective information as shown below:

- City Name
- Manufacturer name of the Product
- Model Number (32-bit max)
- Cost of the product (100000.00 max)
- Month of sale

Write functions to read the data from the CSV file which has the data stored in the text format. Extract the values into an array of structure so that the below operations can be performed.

- Find out total amount of sales in the year.
- Find out total the month with minimum amount of sales.
- Find out total the month with maximum amount of sales.
- Find out details of all products sold in given month.
- Find out details of all products sold which are manufactured by a given manufacturer.
- Find out details of all product sold in a given month for given manufacturer.
- Find out details of all sales from all manufacturers in a given city.
- Sort the sales in ascending order based on the cost of the product.

Note:

- **Use dynamic memory allocation for arrays and structures.**
- **Return appropriate results for all the functions instead of printing.**
- **Assume minimum 5 cities and minimum 10 different manufacturers with minimum 5 different products from each.**

2. Guidelines:

Implement the solution in a step by step manner and test the functions for the expected output.

a. Follow Modular programming and multifile approach
b. Create Makefiles for compiling and testing(minimum). Can add any other options if required.
c. Create libraries for the source code written, link and test the same.
d. Follow best practices while writing the code.

3. Assessment Metrix:

Objective	Description	Marks
• Logic & Expected O/P	• Program should compile without errors. • Should give expected output for supplied input • No compile time warnings • **No scanf usage in business logic**	20
• Error handling, Modularity • Readability, Comments • Portability	• NULL check for pointers in functions. • Function & Variable naming should be self-explanatory • Code style and uniform naming convention (snake case or camel case or any other) • Comments for file and functions. • Header files and source files should be used. • Header files must contain Header guards. • Functions must be used instead of writing everything under main function. • Pass pointer parameters to input data and result data where returning of large data is required. • Avoid accessing Global variable in functions.	20
• Resource efficiency • Time efficiency • Safe parameter passing and return values	• Logic expected with optimal complexity, no redundant logic • Use const keyword whenever passed input prams are not changed • Functions must not return local pointers, structures. • Pass pointers to result in case of structure or array. • Dynamically allocated memory must be freed. • No local statically allocated arrays with big size. • Pass by value should not be used for structures.	20
• Unit Testing and coverage	• Writing Unit test cases for all the functions. • Testing all the corner cases and error conditions. • Code coverage using gcov.	20
• Tool Usage	• Makefile usage. • Less static analysis errors and heap errors (cppcheck, Valgrind etc.) • Creation of libraries for the written code. • Doxygen based documentation	20
Total		**100**