### System Structure Description

* **Company**: Serves as the main entity, holding information about the company name, headquarters location, and a list of its factories, stores, and vehicle models.
* **Car Model**: Describes individual car models manufactured by the company, including details such as model name, specifications, available components, and the manufacturing plant identifier.
* **Worker**: Represents company employees, with details such as employee ID, name, role, and the factory or store they are assigned to.
* **Factory**: Describes the company's production sites, including factory ID, location, production capacity, and a list of vehicle models manufactured there.
* **Car Component**: Specifies the components used in vehicle models, such as engine types, transmissions, and interior features. It includes component ID, name, specifications, and price.
* **Shop**: Represents dealership stores that sell the company’s vehicles, containing information about store ID, location, vehicle model inventory, and sales data.
* **Distribution Network**: Focuses on the company’s logistics and supply chain, detailing network ID, transportation types, regions, and distribution routes.

### Data Compression into Binary Format

All employee data will be compressed into a binary file as follows:

* We assume that employee names contain only letters, allowing them to be stored using 5 bits per character, with a maximum length of 256 characters.
* We assume the company has no more than 65,536 employees, so 16 bits are sufficient for employee IDs.
* Monthly salaries do not exceed one million shekels and are divisible by 1,000, so we use 16 bits to store the salary after removing the last three zeros.
* The company has no more than 256 factories and stores.
* No assumptions were made regarding department names or lengths to allow flexibility in naming factories and other entities.

### Employee Representation in Binary Format

| **Field** | **Size** |
| --- | --- |
| Role | 1 byte \* amount |
| Department Name | Variable Length |
| Department Name Length | 1 byte |
| Factory ID | 4 bytes |
| Salary | 2 bytes |
| Employee ID | 2 bytes |
| Employee Name | (Name Length \* 5 bits) |
| Name Length | 1 byte |

### User Operations

* **View and Edit Factory Data** – Allows access to details about each factory in the company.
* **Add New Vehicle Models** – Enables the addition of a new vehicle model to the company's catalog, including details such as model name, technical specifications, and price.
* **Add Employees** – Adds an employee to the system and assigns them to the relevant factory or store.
* **Add Factories** – Adds a factory and aligns it with the appropriate distribution network.
* **Add Stores** – Adds a store and aligns it with the appropriate distribution network.
* **Add Vehicle Components** – Adds a component to a factory that manufactures it.
* **Add a New Vehicle Model** – Adds a new model to a factory based on relevant components.
* **Sort Employees by Name, ID, or Work Details** – Ability to sort stores, factories, or employees based on various criteria such as sales, number of employees, or alphabetically.
* **Search for an Employee** – Searches the system to find specific details about employees.
* **Fire an Employee** – Allows the company to remove an employee from the sys

**System Diagram**

char\* company\_name;

ShopFactoryNode\* head\_shops\_and\_factories // nested list

Worker\* CEO;

int num\_shops\_and\_fact; // Total number of shops and factories

Worker\*\* workers; // Array of workers

int num\_workers;

int max\_workers;

int id\_counter;

DistributionNetwork south, north; // Distribution networks

void init\_company(Company\* company, const char\* name);

void change\_CEO(Company\* company, Worker\* ceo);

void add\_shop\_factory\_node(Company\* company, void\* shop\_or\_factory, int is\_shop);

Worker\* add\_worker(Company\* company, const char\* name, const char\* position, double salary, const char\* department, int factory\_id);

ShopFactoryNode\* find\_factory\_shop\_by\_id(Company\* company, int factory\_id);

CarModel\* find\_model\_by\_name(Company\* company, const char\* model\_name);

int find\_worker\_by\_id(Company\* company, int worker\_id);

void free\_company(Company\* company);

void move\_worker(Company\* company, int worker\_id, int factory\_shop\_id);

void fire\_worker(Company\* company, int worker\_id);

void display\_workers\_company(Company\* company);

Struct CarModel:

Char\* model\_name[30]

Int model\_year

Char\* manufacturer

Double price

CarComponent\*\* components;

int num\_components

Int production\_volume

Functions:

Void update\_model\_name(CarModel\* model, Char\* new\_name);

Void update\_model\_year(CarModel\* model, Int new\_year);

Void update\_manufacturer(CarModel\* model, Char\* new\_manufacturer);

Void update\_price(CarModel\* model, Double new\_price);

Void add\_component(CarModel\* model, CarComponent\* component);

Void remove\_component(CarModel\* model, Int componentIndex);

Int calculate\_production\_cost(CarModel\* model);

Void display\_model\_info(CarModel\* model);

Struct Shop:

Int shop\_id;

Char\* location

Car\_Models\*\* inventory

Double sales

Worker\*\* workers

Functions:

Void update\_location(Shop\* shop, Char\* new\_location)

Void add\_to\_inventory(Shop\* shop, Car\_Models\* model)

Void remove\_from\_inventory(Shop\* shop, Int modelIndex)

Void update\_sales(Shop\* shop, Double sales\_amount)

Void display\_shop\_info(Shop\* shop)

Struct DistributionNetwork:

Int network\_id

Char\* description

Factory\*\* source\_factory

Shop\*\* destination\_shops

Int routes\_count;

Functions:

Void update\_description(DistributionNetwork\* network, Char\* new\_description);

Void add\_destination\_shop(DistributionNetwork\* network, Shop\* shop);

Void remove\_destination\_shop(DistributionNetwork\* network, Int shopIndex);

Void update\_routes\_count(DistributionNetwork\* network, Int new\_count);

Void display\_network\_info(DistributionNetwork\* network);

Struct Factory:

Int factory\_id

Char\* location

Int production\_capacity

Car\_Models\* produced\_models

CarComponent\* produced\_components

Worker\*\* workers

Functions:

Void update\_location(Factory\* factory, Char\* new\_location)

Void update\_production\_capacity(Factory\* factory, Int new\_capacity)

Void add\_produced\_model(Factory\* factory, Car\_Models\* model)

Void remove\_produced\_model(Factory\* factory, Int modelIndex)

Void display\_factory\_info(Factory\* factory)

Struct Worker:

Char\* worker\_name

Int worker\_id

Char\* position

Double salary

Char\* department

Int factory\_id

Functions:

Void update\_worker\_name(Worker\* worker, Char\* new\_name);

Void update\_position(Worker\* worker, Char\* new\_position);

Void update\_salary(Worker\* worker, Double new\_salary);

Void assign\_to\_department(Worker\* worker, Char\* new\_department);

Void assign\_to\_factory(Worker\* worker, Int new\_factory\_id);

Void display\_worker\_info(Worker\* worker);

Struct CarComponent:

Int component\_id

Char\* name

Char\* type

Double cost

Functions:

Void update\_component\_name(CarComponent\* component, Char\* new\_name);

Void update\_component\_type(CarComponent\* component, Char\* new\_type);

Void update\_cost(CarComponent\* component, Double new\_cost);

Void display\_component\_info(CarComponent\* component);

**System Functions**

// Upload system data from file

void upload\_system\_data(Company\*\* company);

// write system data to file

void write\_system\_data(Company\* company);

// Start the company system

Company\* start\_company\_system();

void run\_company\_system(Company\*\* company);

// Display functions

void display\_company(Company\* company);

void display\_distribution\_networks(Company\* company);

void display\_shops(Company\* company);

void display\_factories(Company\* company);

void display\_car\_models(Company\* company);

// Add new entities

void change\_CEO\_to\_sys(Company\* company);

void add\_worker\_to\_sys(Company\* company); // Adds a new worker based on user input

void add\_shop\_to\_sys(Company\* company); // Adds a new shop based on user input

void add\_factory\_to\_sys(Company\* company); // Adds a new factory based on user input

void add\_car\_model\_to\_sys(Company\* company); // Adds a new car model based on user input

void add\_component\_to\_sys(Company\* company);

void add\_model\_to\_inventory\_sys(Company\* company);

// Sorting workers

void sort\_workers\_by\_name(Company\* company);

void sort\_workers\_by\_id(Company\* company);

void sort\_workers\_by\_job\_details(Company\* company);

// Searching workers

void search\_worker\_by\_name(Company\* company);

void search\_worker\_by\_id(Company\* company);

void search\_worker\_by\_job\_detail(Company\* company);

// move worker from factory/shop to factory/shop

void move\_worker\_factory\_shop(Company\* company);

// Fire worker from the company

void fire\_worker\_from\_sys(Company\* company);