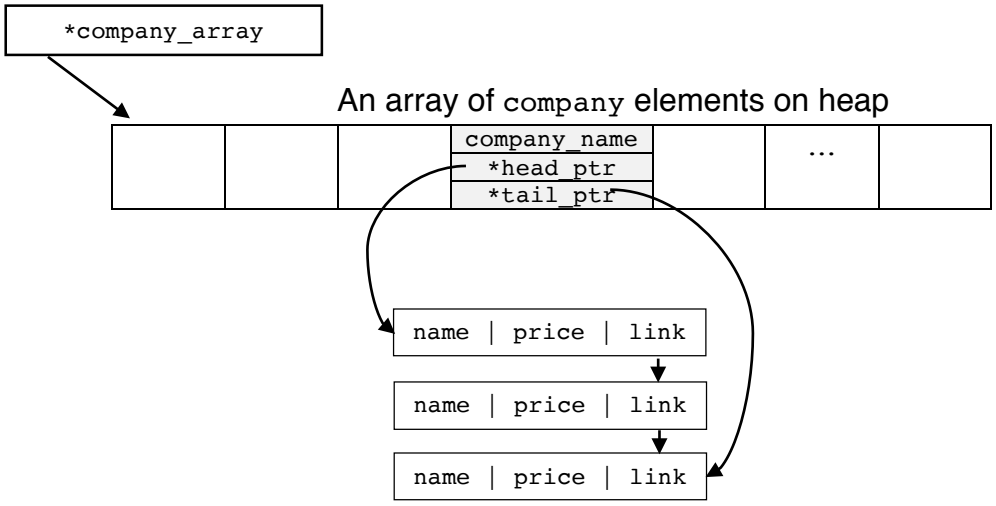
COEN 79L - Object-Oriented Programming and Advanced Data Structures

Lab 7

- **Database**



A database class to store the name of companies as well as their products information.

The database class is implemented as a dynamic array which holds `company` objects. Therefore, the private variables of the database class are:

```
1. company* company_array; // A pointer to a dynamic array of company objects
2. size_type aloc_slots; // Number of slots in the stindex_array array
3. size_type used_slots; // Number of used slots in the stindex_array array
```

Each `company` object has three member variables:

```
1. std::string company_name; // Name of company
2. node* head_ptr; // Head of the linked list including company's products
3. node* tail_ptr; // Tail of the linked list including company's products
```

The `head_ptr` and `tail_ptr` are pointers to the start and end of a linked list. This linked list is used to store the products of each company.
To this end, we define the `node` class with the following member variables:

```
1. std::string name; // Name of the product
2. float price; // Price of the products
3. node * link; // Link to the next product
```

Initially, the size of `company_array` is 2 (i.e., a dynamic array with two slots). When a `company` is added, and the current array is full, an extra slot is allocated using the `reserve`

function. A new pointer to an array is then created and then the contents from the older array are copied into the new array.

The class enables us to add companies and their products to the database. However, note that no company or product duplicate is allowed. You should implement functions to check these conditions.

Every product insertion results in adding a new node to the linked list of that company. Please note that a product cannot be added if the company does not exist in the database.

When a company is deleted, first the linked list is erased (as it is on heap memory) and then the dynamic array of the database class is updated to reflect the company removal. This may require shifting the elements of this array.
Note that you should implement destructors for the company class and database class. The destructor of the company class clears the linked list of that company. The destructor of the database class deletes the dynamic array of companies.

Please note that proper operation of the `database` class requires the implementation of constructor, copy constructor, assignment operator, and destructor, for the `database` class and `company` class. For example, when you assign a `database a` to a `database b`, you should ensure that in addition to the dynamic arrays of `companies`, the linked lists are copied as well. Otherwise, the entries of the dynamic array of `b` point to the linked lists of `a`, which is not what we want.

Given files:

- `database.h and database.cpp` - Incomplete implementation of the `database` class.
- `company.h company.cpp` - Incomplete implementation of the `company` class.
- `node.h` and `node.cpp` – Node class for the products stored in linked list.
- Test files. (feel free to extend the test files to verify all the operations.)