

Lab 3: Linked List - Stack - Queue

1 Hash Table

1.1 Content

The tax code information in this lab is collected from 1250 Vietnam company. You can find it in *"MST.txt"*, which has the content as follow:

```

1  Ten cong ty|MST|Dia chi
2  CONG TY TNHH BEE VIET NAM|0108927262|So 8 - K8, Khu nha o lien ke trung tam 75, Tong cuc II, Bo Quoc Phong, thon Lai Xa, Xa Kim Chung, Huyen
   Hoaí Duc, Thanh pho Ha Noi
3  CONG TY CO PHAN THUONG MAI CHAU DUC PHAT|3502406778|So 266 Ap Phuoc Trung, Xa Tam Phuoc, Huyen Long Dien, Tinh Ba Ria - Vung Tau
4  CONG TY CO PHAN XAY DUNG DAU TU PHAT TRIEN DI SAN SAO VIET|0315938079|30/18 Truong Sa, Phuong 17, Quan Binh Thanh, Thanh pho Ho Chi Minh
5  CONG TY TNHH MTV THAN TAN HOANG LONG|0315938103|2/47 Duong Thanh Loc 31, Khu Pho 3C, Phuong Thanh Loc, Quan 12, Thanh pho Ho Chi Minh
6  CONG TY TNHH NONG NGHIEP CONG NGHE CAO MIEN DONG VIET|3401194911|Thon 5, Xa Tan Phuoc, Huyen Ham Tan, Tinh Binh Thuan
7  CONG TY TNHH XAY DUNG VINH GIA PHAT|3603671412|So 171, Xom 4, Khu 2, Ap Bau Ca, Xa Trung Hoa, Huyen Trang Bom, Tinh Dong Nai
8  CONG TY TNHH THUONG MAI DICH VU PHU LONG RIVERSIDE|3401194929|243 Huynh Thuc Khang, KP1, Phuong Mui Ne, Thanh pho Phan Thiet, Tinh Binh Thuan
9  CONG TY TNHH HANH TRANG PHAT|3603671155|To 5, Ap Thanh Binh, Xa Loc An, Huyen Long Thanh, Tinh Dong Nai
10 CONG TY TNHH THUONG MAI DICH VU THIET BI M.K.K|0315932380|154/1/34 Cong Lo, Phuong 15, Quan Tan Binh, Thanh pho Ho Chi Minh
11 CONG TY TNHH TM - DV - NHA HANG HAI SAN KY QUANG|0315933352|So 526 Duong Pham Van Dong, Phuong 13, Quan Binh Thanh, Thanh pho Ho Chi Minh
12 CONG TY TNHH MTV KIM LONG|1201613551|So 170 Nguyen Minh Duong, Ap 1, Xa Dao Thanh, Thanh pho My Tho, Tinh Tien Giang
13 CONG TY TNHH SONA AGENCY VIET NAM|0108926660|So 333 Bach Mai, Phuong Bach Mai, Quan Hai Ba Trung, Thanh pho Ha Noi

```

in which:

- The first line provides the included information fields.
- For the next lines, each one is the information of 1 company, separated by a straight dash (|).

For this lab, students are required to read the info of Companies from the *"MST.txt"* file into the **Company** data structure, and store as a hash table.

1.2 Programming

The **Company** data structure is defined as follow:

```

struct Company
{
    string name;
    string profit_tax;
    string address;
};

```

Fulfill the following requirements:

1. Read the companies information from a given file:
 - `vector<Company> ReadCompanyList(string file_name)`
 - Input: `file_name` direction to the input file (*"MST.txt"* for this lab).
 - Output: Companies list extracted from the file, which has the data type `vector<Company>`.
2. Hash a string (company name) function:
 - `long long HashString(string company_name)`
 - Input: `company_name` is the string (company name), that need to be hashed.
 - Output: `long long` positive integer, result of the hash formula given below.

- Hash formula:

$$hash(s) = \left(\sum_{i=0}^{n-1} (s[i] \times p^i) \right) \bmod m$$

in which:

- **s** Last 20 characters of the **company_name**. The whole string is required if its size doesn't exceed 20.
- **s[i]** ASCII code of the character at position **i** from **s**.
- $p = 31$
- $m = 10^9 + 9$

3. The function to create a hash table of size 2000, generated from the Companies list:

- **Company* CreateHashTable(vector<Company> list_company)**
- Input: **list_company** Companies list extracted from file.
- Output: Generated hash table.
- Note: Linear probing is required.

4. Add the info of 1 company into an existed hash table:

- **void Insert(Company* hash_table, Company company)**
- Input: - **hash_table**: Given hash table.
- **company** the string name of the company, which need to be hashed.

5. Search for company information by its name:

- **Company* Search(Company* hash_table, string company_name)**
- Input: - **hash_table** Given hash table.
- **company_name** the string name of the company, which data is needed.
- Output: Information of the required company, store as **Company** data structure. Return NULL if the company cannot be found.

2 Doubly Linked list

Following is representation of a doubly linked list:

```
struct d_NODE{
    int key;
    d_NODE* pNext;
    d_NODE* pPrev;
};
```

```
struct d_List{
    d_NODE* pHead;
    d_NODE* pTail;
};
```

Implement functions to execute the following operations:

1. Initialize a list from a given integer.
2. Add a node at the front of a given List.
3. Add a node before a given node.
4. Add a node after a given node.
5. Add a node at the end of a given List.
6. Remove a node at the front of a given List.
7. Remove a node at the end of a given List.
8. Remove the first node with given value.