Full name: Mai Anh Phúc Minh

Student ID: 24120094

Report week 6.

No.	Percentage understood	Content understood	Percentage Referenced	Content Referenced	Referenced Source
1. readCompanyList()	100%				
2. hashString()	100%				
3. createHashTable()	100%				
4. insert()	100%				
5. search()	100%				
6. main()	100%				

Em xin được báo cáo về bài tập tuần 6. Với yêu cầu sử dụng cấu trúc dữ liệu Hash Table để chia các công ty và thực hành search thì em đã thiết kế **3 hàm phụ** và **5 hàm chính** để xử lí yêu cầu.

1. **Các hàm phụ.**
   - modPower(): được xây dựng để tính lũy thừa và lấy modulo cùng lúc để tránh việc tràn số. Sử dụng phương pháp phân tích số mũ thành hệ nhị phân và tính chất (a.b) mod c = (a mod c)(b mod c) thì ta tránh được việc số sẽ bị tràn, ngoài ra thay vì có độ phức tạp là O(n) thì thay vào đó ta sẽ có được O(log n) giúp cải thiện hiệu suất tính.

   - createCompany(): được xây dựng để tạo một con trỏ Company phục vụ cho việc tạo ô trong Hash Table. Khi ta đưa vào tham số là một company thì nó sẽ trả về cho ta một con trỏ tương ứng.

   - deleteHashTable(): được xây dựng để xóa Hash Table mà ta đã tạo để tránh việc tràn bộ nhớ sau khi thực thi đoạn code. Ta chỉ cần duyệt qua mọi vị trí trong bảng và xóa những vị trí chứa các company là được.

2. **Các hàm chính.**
   - readCompanyList(): được xây dựng để đọc các company từ file MST.txt và trả về một vector<Company>. Khi ta mở được file txt thì ta tạo một stream gồm mỗi dòng trong file, chứa các thông tin của 1 company, sau đó ta tách các dấu

‘|’ ra là sẽ có được các thông tin cần thiết. Cuối cùng là push back vào vector mà ta khởi tạo và trả về vector đó sau khi đã đọc được hết các dòng trong file.

- hashString(): đươc khởi tạo để trả về vị trí của company đó trong Hash Table. Dựa trên công thức hàm băm có sẵn, ta cần chú ý điều kiện lấy 20 kí tự cuối để xét và tính toán, duyệt qua từng kí tự và nhân nó cho một hằng số được lũy thừa dựa trên vị trí của kí tự, cuối cùng là lấy modulo cho 2000.

- createHashTable(): được xây dựng để tạo nên một Hash Table từ một vector có sẵn. Ta duyệt qua mọi phần tử có trong vector, sau đó lấy name của company đó và tính vị trí trong bảng dựa vào hàm băm được nêu ở trên. Nếu có xảy ra đụng độ, 2 kết quả băm giống nhau cho 2 company khác nhau, thì ta sẽ giải quyết theo linear probing. Ta tăng vị trí cần xét lên 1 đơn vị và xét tiếp xem vị trí đó có trống không, nếu có thì sẽ đưa company vào vị trí đó còn không sẽ tiếp tục cộng dồn lên 1 vị trí cho tới khi tìm được vị trí thỏa mãn. Cuối cùng sẽ trả về một Hash Table hoàn chỉnh.

- insert(): được xây dựng để thêm 1 company mới vào trong Hash Table có sẵn. Tương tự như việc khởi tạo, ta sẽ lấy tên của company đưa vào hàm băm để tìm vị trí trong bảng, giải quyết đụng độ bằng linear probing, tuy nhiên, vì đây là hàm thêm vào nên sẽ có một biến đánh dấu vị trí khởi điểm, nếu như sau khi cộng dồn nhiều lần và quay lại vị trí bắt đầu thì ta sẽ trả về kết quả là bảng đã đầy.

- search(): được xây dựng để tìm kiếm 1 company trong Hash Table và trả về con trỏ đang trỏ về company đó. Ta lấy tên của company và đưa vào hàm băm để lấy được vị trí xuất phát, lưu nó thành một biến và bắt đầu việc tìm kiếm, nếu như không phải company mình cần tìm thì sẽ tăng vị trí lên 1 và xét tiếp xem có phải không cho tới khi nào tìm được thì sẽ trả về con trỏ ở vị trí đó hoặc tới khi ta quay về lại vị trí xuất phát hoặc gặp một ô trống thì đồng nghĩa với việc ta không tìm được và trả về nullptr.

- main(): được xây dựng để mở cái file nhập và xuất với các tham số đưa vào là tên các tập tin cùng trong thư mục. Sau khi mở được các tệp thì sẽ đưa các company trong MST.txt vào một vector để chuyển hóa nó thành một Hash Table để lưu trữ các company đó. Rồi sẽ duyệt qua các dòng của file input để

lấy được tên các company cần tìm, rồi đưa vào file output các thông tin theo mẫu xuất có sẵn.

3. Minh chứng Github.