**CPP.Assignment 12 – Advance features in C++**

**Exercise 1+2:**

Áp dụng phương pháp lập trình hướng đối tượng OOP (mà chủ yếu ở đây là bảo vệ dữ liệu), thay vì sử dụng struct, em sử dụng class để xây dựng lớp Student và lớp DouLinkedList (doubly linked list), việc sử dụng class, đặc biệt là cho lớp DouLinkedList có một vài ưu việt so với sử dụng struct:

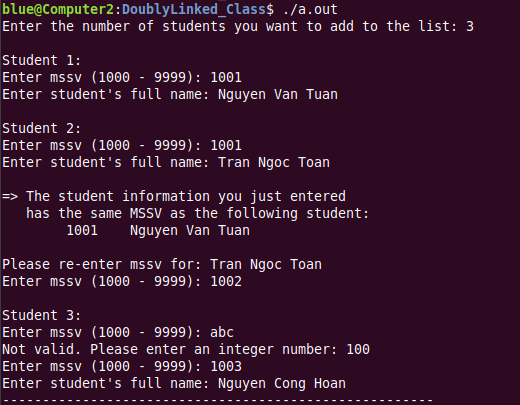
* **Bảo vệ dữ liệu (head, tail và các node ở giữa)** gần như hoàn toàn (em đã cố gắng hết mức, nhưng do kinh nghiệm em còn hạn chế nên em cũng không chắc chắn còn nỗ hổng nào không). Như hầu hết các phương thức trả về kiểu void, chỉ có 2 phương thức phương thức trả về Node\* (em đã khống chế thành kiểu **const** Node\*). So với việc sử dụng struct, chưa nói đến có thể dễ dàng truy cập vào các nodes từ head, hoặc tail, mà nếu thao tác không cẩn thận rất dễ để mất head hoặc tail.
* **Tự động giải phóng cấp phát động bằng destructor** khi list thoát khỏi phạm vi khai báo (em học được từ mart\_pointer). Bởi vì khi thêm node vào list, chúng ta luôn cấp phát động cho node mới sử dụng toán tử new, hơn nữa data trong list em sử dụng con trỏ (cấp phát động data, một phần để linh hoạt, một phần để swap data sẽ dễ dàng hơn swap các node không cần thiết). Sẽ rất khó để nhớ giải phóng cả các node lẫn data dễ dẫn tới rò rỉ memory.

Chương trình đưa ra 5 tính năng: nhập, in ra, xóa, tìm kiếm và sắp xếp có thể chọn để sử dụng liên tục.

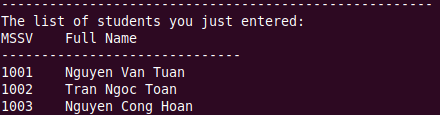
Sau đây là kết quả output của các yêu cầu:

* **Input**: Nhập sinh viên từ console (sử dụng phương thức add\_tail)

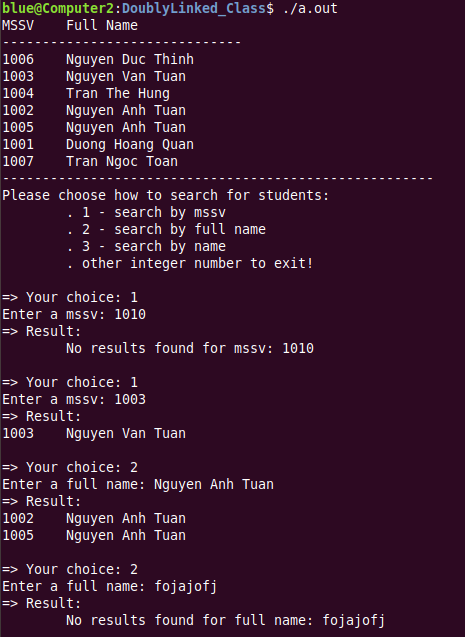
Không cho nhập trùng mssv như yêu cầu, ngoài ra em hạn chế mssv phải là số nguyên không khoảng 1000 – 9999. Họ và tên đầy đủ được bỏ đi những khoảng trắng ở đầu và cuối nếu có.

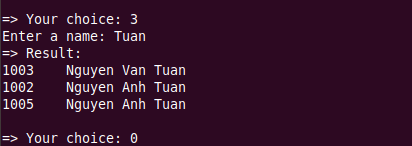


* **Show**: In danh sách đã nhập ra console (từ head tới tail: theo chiều nhập - sử dụng phương thức print\_down)

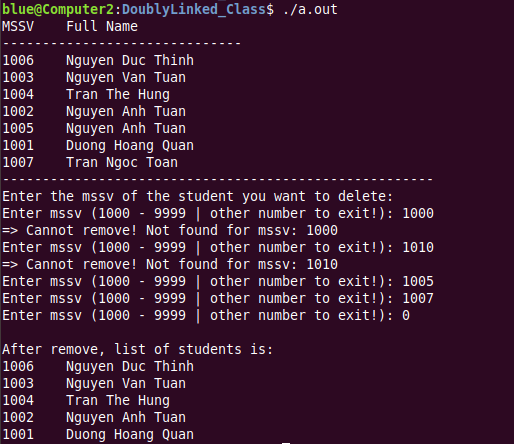


* **Search**: In ra danh sách sinh viên và test tìm kiếm theo 3 tiêu chí đề ra: lần lượt theo mssv, họ và tên, tên.

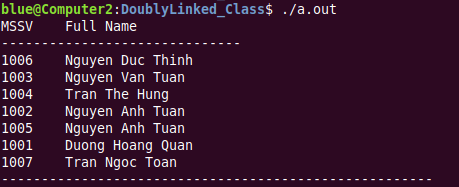




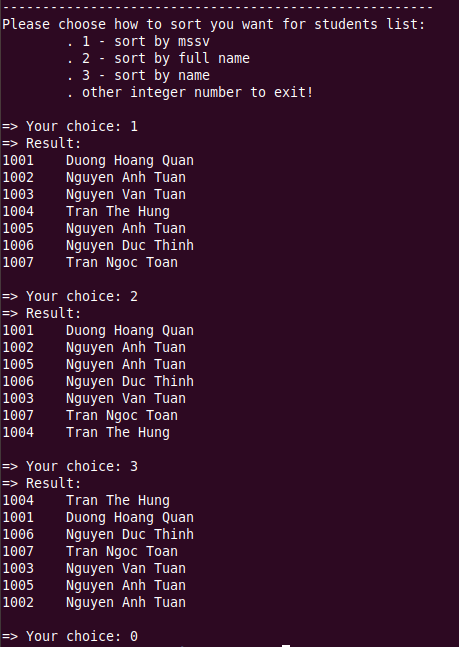
* **Remove**: In ra danh sách sinh viên, thực hiện xóa sinh viên theo mssv, in lại danh sách sau khi xóa ra console:



* **Sort**: In ra danh sách trước khi sắp xếp và test sắp xếp cho 3 tiêu chí đề ra:



Kết quả sort: Lần lượt theo mã số sinh viên, họ và tên, tên (chiều tăng dần)



=> Kết quả test memory với valgrind cho thấy, thiết kế class với destructor hoạt động hiệu quả, sau khi chạy qua tất cả tính năng, kết quả hoàn toàn không có rò rỉ bộ nhớ.

