ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

NHẬP MÔN LẬP TRÌNH

Thực hành

BÀI THI CUỐI KỲ - CA 2

Giảng viên thực hành: Lê Đức Khoan

Thành phố Hồ Chí Minh, 05/2025

Mục lục

1 Yêu cầu bài nộp

Sinh viên được cung cấp một thư mục mã nguồn **final_s2** chứa thư viện, các định nghĩa hàm. Mã nguồn đã bao gồm đầy đủ các định nghĩa hàm. Sinh viên thực hiện thực các hàm theo prototype đã được định nghĩa ở file **.h** và có thể viết thêm hàm nếu cần thiết.

Chú ý: Sinh viên không được tự ý sửa đổi bất kỳ thành phần nào của mã nguồn mẫu. Nếu mã nguồn biên dịch không thành công khi chấm thì sẽ nhận điểm 0.

Khi nộp bài sinh viên **đổi tên thư mục final_s2 thành MSSV và zip toàn bộ mã nguồn** thành file **MSSV.zip** với MSSV là mã số sinh viên được nhà trường cung cấp. Các thành phần ở trong thư mục phải giống như thư mục được cung cấp ban đầu. Không thêm bất kỳ file hay thư mục nào khác.

Tên file zip mẫu:

```
24123456.zip
```

Tất cả các trường hợp làm sai yêu cầu sẽ nhận điểm 0 cho bài thi.

2 Hướng dẫn chạy file thực hành cuối kỳ

Tổ chức thư mục của final_s2 như sau:

```
final_s2
   |-- restaurant_helper
   |           |-- restaurant_utils.h
   |           |-- restaurant_utils.cpp
   |           |-- data
   |           |     |-- restaurants.txt
   |           |-- result
   |                 |-- quality_restaurants.txt
   |-- main.cpp
   |-- Makefile
```

Vì có nhiều file được include vào trong file main để chạy chương trình do đó để nhanh chóng compile và chạy chương trình. Trong bài lab này chúng ta sẽ tiếp cận với giải pháp dùng Makefile để compile và chạy chương trình. Gõ vào terminal để kiểm tra version của Make:

```
make -v
```

Nếu không xuất hiện version thì máy chưa có Make. Khi đó, các bạn có thể tham khảo hướng dẫn cài đặt Make. Sau khi đã cài xong, các bạn đi đến thư mục đang chưa file **Makefile** và có thể thực hiện các câu lệnh sau:

- **make all**: Compile code ở tất cả các file để tạo ra file thực thi (**.exe**). Đây là default command nên nếu bạn gõ **make** thì nó sẽ tự động hiểu là **make all**.
- **make clean**: Để xoá bỏ file thực thi vừa tạo.
- **make run**: Thực hiện compile code và chạy chương trình.

Các câu lệnh trong Makefile bao gồm:

```
1  # Compiler and flags
2  CXX = g++
3  CXXFLAGS = -std=c++17 -Wall -g  -I restaurant_helper
4
5  # Name of execution file
6  TARGET = main
7
8  # List of source files
9  SRC = main.cpp restaurant_helper/restaurant_utils.cpp
10
11 # Default command
12 all:
13         $(CXX) $(CXXFLAGS) $(SRC) -o $(TARGET)
14
15 # Clean command
16 clean:
17         rm -f $(TARGET)
18
19 # Compile and run command
20 run: all
21         ./$(TARGET)
```

- **Câu lệnh số 2**: Khai báo compiler là **g++**.
- **Câu lệnh số 3**: Khai báo thêm các flags khi compile code. Với các tham số sau **-I** chỉ các thư mục chưa file **.h**. Nếu có thêm nhiều thư viện tự định nghĩa ta sẽ thêm tiếp các cặp **-I library_directory**. Mỗi thư mục sẽ đi với một ký tự **-I**.
- **Câu lệnh số 6**: Khai báo tên file **Thực thi** được tạo ra khi compile mã nguồn.
- **Câu lệnh số 12**: Compile mã nguồn và tạo ra file thực thi. Với câu lệnh tương tự trong những tuần trước: **g++ main.cpp -o main**. Câu lệnh này sẽ được chạy khi ta gọi: **make all** hoặc **make**.

- **Câu lệnh số 16**: Xoá file thực thi vừa được tạo ra. Lệnh được thực thi khi gọi: **make clean**.
- **Câu lệnh số 20**: Thực hiện compile và chạy file thực thi. Lệnh được thực thi khi gọi: **make run**. Khi gọi lệnh này sẽ thực hiện hai việc liên tục là **make all** và **./main**. Với **make all** đã được trình bày ở trên và **./main** là để chạy file thực thi.

3 Đề thi

3.1 Nhà hàng chất lượng - (6 điểm)

Một hệ thống đánh giá nhà hàng theo điểm số của các món ăn được lưu trong một ma trận kích thước $m \times n$, trong đó:

- Mỗi hàng là một nhà hàng.
- Mỗi cột là điểm số cho mỗi món ăn.

Một nhà hàng được gọi là **nổi tiếng** ở một món ăn nếu điểm đánh giá món ăn đó thoả mãn:

- Là **cao nhất** trong các món ăn của nhà hàng đó (trong hàng).
- Và đồng thời là **cao nhất** trong món ăn đó (trong cột).

Điểm cao nhất phải là duy nhất trong hàng và cột.

Yêu cầu bài tập: Hiện thực hàm `find_quality_restaurants` để tìm tất cả các nhà hàng có món ăn nổi tiếng. Đầu vào và đầu ra của bài toán sẽ được ghi ra file với:

- Đầu vào sẽ được đọc từ file **restaurants.txt**
- Đầu ra sẽ được ghi vào file **quality_restaurants.txt**.

Chú ý: Sinh viên tự hiện thực các các hàm phụ để giải quyết bài toán và chỉ được hiện thực trong file **.cpp**, không được thêm định nghĩa hàm vào file **.h**.

```
void find_quality_restaurants(input_file_name = "restaurants.txt",
                              output_file_name = "quality_restaurants.txt") {
    // TODO

    // END TODO
}
```

Testcase mẫu:

```
// restaurants.txt
2
```

```
3
Restaurant A,8.0,9.0,7.0,6.0
Restaurant B,5.5,6.0,7.5,7.5
Restaurant C,6.0,5.0,6.0,7.5
4
Restaurant A,8.0,9.0,7.0,6.0
Restaurant B,5.5,6.0,7.5,7.5
Restaurant C,6.0,5.0,9.0,7.5
Restaurant D,6.0,7.5,8.5,9.5
```

Giải thích file đầu vào:

- Dòng đầu tiên: Số lượng testcase trong file.
- Dòng thứ 2: Số lượng nhà hàng của testcase 1.
- 3 dòng kế tiếp: Thông tin của nhà hàng bao gồm tên nhà hàng, và điểm 4 món ăn (Chỉ có 4 món ăn). Các phần tử cách nhau bằng dấu ",".
- Dòng thứ 6: Số lượng nhà hàng của testcase 2.
- 4 dòng kế tiếp: Thông tin của nhà hàng.

Kết quả:

```
// quality_restaurants.txt
Restaurant A,9.0
Restaurant A,9.0
Restaurant C,9.0
Restaurant D,9.5
```

Giải thích kết quả:

- Ở testcase đầu tiên chỉ có nhà hàng A thoã mãn yêu cầu.
- Ở testcase thứ 2 có 3 nhà hàng là A, C, D thoã mãn.

Chú ý: Ghi vào file đầu ra tên nhà hàng và điểm số thoã mãn yêu cầu (cách nhau bởi dấu ",").

3.2 Sinh viên duy nhất - (4 điểm)

Một hệ thống ghi nhận mã số sinh viên đăng ký các khóa học trực tuyến trong một học kỳ. Mỗi sinh viên có thể đăng ký nhiều khoá học. Nhà trường cần tìm kiếm những sinh viên đăng ký chỉ duy nhất một môn học để đánh giá tình trạng học tập. Do đó, hãy tìm các mã sinh viên chỉ xuất hiện đúng một lần trong danh sách đăng ký — tức là sinh viên chỉ tham gia duy nhất một khóa.

Hiện thực hàm `find_unique_students` để thực hiện yêu cầu trên.

```
int find_unique_students(int ids[], int n) {
    // TODO

    // END TODO
    return 0;
}
```

Testcase mẫu:

```
Input: [2001, 2002, 2003, 2001, 2004, 2002]
Output: 2
```

Giải thích:

- Chỉ có sinh viên 2003 và 2004 tham gia một khoá học → có hai sinh viên tham gia duy nhất một khoá học.

Output là giá trị hàm trả về, không phải là kết quả in ra màn hình.