ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

Bài tập lớn 1

RESTAURANT OPERATIONS
(Phần 1)

TP. HỒ CHÍ MINH, THÁNG 02/2023

Đặc Tả Bài Tập Lớn 1

Phiên bản 1.0

1 Chuẩn đầu ra

Sau khi hoàn thành bài tập lớn này, sinh viên sẽ có khả năng:

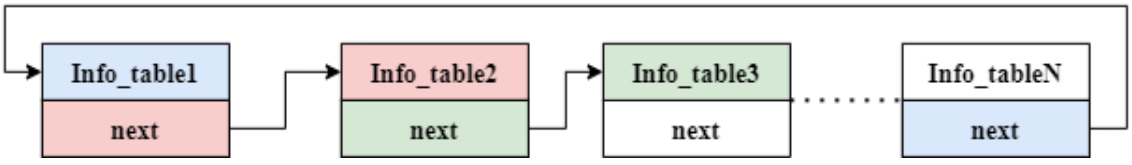
- Hiện thực cấu trúc dữ liệu danh sách liên kết vòng.
- Hiện thực các thao tác cơ bản trên kiểu dữ liệu danh sách liên kết vòng.

2 Giới thiệu

Trong bài tập lớn này, các em sẽ mô phỏng việc xử lý yêu cầu đặt bàn và sắp xếp vị trí chỗ ngồi cho khách hàng trong một nhà hàng thông qua các lệnh được mô tả ở phần 2.2. Quy trình vận hành tại nhà hàng như sau:

1. Đầu tiên, khách hàng có thể gọi điện đặt bàn trước hoặc đến trực tiếp nhà hàng để nhân viên sắp xếp chỗ ngồi. Khách hàng có thể chọn vị trí chỗ ngồi theo ý thích hoặc để nhân viên xếp chỗ tùy ý.
2. Sau khi một khách hàng đã dùng xong món và hoàn tất thanh toán, nhân viên sẽ mời một vị khách khác ngồi vào bàn trên để dùng bữa. Nếu nhà hàng đang không có khách thì bàn trên sẽ thành bàn trống để chờ các vị khách tiếp theo.
3. Bên cạnh đó, nhân viên có quyền sắp xếp lại hàng chờ để ưu tiên cho các vị khách cao tuổi khi có yêu cầu từ chủ nhà hàng.
4. Ngoài ra, nhà hàng chỉ nhận khách hàng trong độ tuổi từ 16 tuổi trở lên. Nếu khách hàng chưa đủ tuổi thì nhân viên sẽ mời vị khách đó về.

Vì diện tích nhà hàng tương đối hẹp, cho nên chủ nhà hàng chọn cách bố trí các bàn theo dạng danh sách liên kết vòng (như hình 1).



Hình 1: Biểu diễn cách bố trí bàn của nhà hàng.

Trong đó:

- Info_table: chứa thông tin của việc đặt bàn, bao gồm các thông số ID, NAME và AGE. Các thông số trên và số lượng bàn tối đa mà nhà hàng có thể phục vụ cùng một lúc, được mô tả chi tiết trong mục 2.1.

2.1 Hướng dẫn chung

Mỗi một yêu cầu từ khách hàng, nhân viên hay chủ nhà hàng được biểu thị thông qua các lệnh xử lý dữ liệu. Mỗi lệnh là một dòng trên tập tin thông tin (từ in đậm trong mô tả), bắt đầu bằng một từ khoá và theo sau là các thông số (từ đặt trong dấu < và > hoặc [và] trong mô tả). Giữa lệnh và các thông số cách nhau đúng một khoảng trắng. Không có khoảng trắng nào trước từ khoá lệnh và cũng không có khoảng trắng nào đi sau thông số cuối cùng. Các thông số bắt buộc có trong lệnh sẽ được đặt trong dấu < và >. Và một số thông số có thể tuỳ chọn (có hoặc không có trong lệnh), sẽ được đặt trong dấu [và]. **Khi một lệnh không cung cấp đúng số thông số hoặc không có kiểu thông số đúng như mô tả hoặc có các khoảng trắng không như mô tả thì lệnh sẽ không đươc xử lý và sẽ được bỏ qua**. Ngược lại, lệnh sẽ được xử lý theo mô tả trong mục 2.2.

Ý nghĩa của từ viết tắt và kiểu dữ liệu của các thông số được mô tả như sau:

- **ID**: một số nguyên dương (bắt đầu từ 1) thể hiện chỉ mục của các bàn trong nhà hàng với giá trị tối đa là MAXTABLE (được định nghĩa trong file main.h).
- **NAME**: một chuỗi ký tự liên tục không có khoảng trắng, thể hiện tên của khách hàng.
- **AGE**: một số nguyên dương có giá trị từ 1 đến 115.
- **NUM**: một số nguyên dương với ý nghĩa khác nhau ứng với từng lệnh xử lý khác nhau. Và ứng với mỗi lệnh thì giá trị NUM này sẽ có các khoảng giá trị khác nhau.

Lưu ý: mỗi lệnh cần phải được xử lý với độ phức tạp thời gian không vượt quá qui định (độ phức tạp được mô tả ngay sau nội dung lệnh và được tô xanh), với N là số lượng khách hàng và số lượng bàn trong nhà hàng. Sau khi xử lý xong tất cả các lệnh trong tập tin đầu vào và xuất kết quả ra màn hình, chương trình phải đảm bảo huỷ tất cả các đối tượng dữ liệu được cấp phát động, không để lại rác trong bộ nhớ trước khi kết thúc chương trình.

2.2 Danh sách lệnh - Độ phức tạp thuật toán

REG [ID] <NAME> <AGE> - O(N):
Lệnh này dùng để đặt bàn cho khách theo thông tin mà khách hàng cung cấp. Quy trình xử lý lệnh đặt bàn được diễn ra như sau:

- Nếu khách hàng đặt bàn theo ID mà bàn đó còn trống, thì cập nhật thông tin của khách ứng với bàn có ID.
- Trường hợp khách hàng muốn nhân viên xếp bàn tùy ý (lệnh đặt không có thông số ID) thì nhân viên sẽ chọn bàn trống đầu tiên tính theo ID từ nhỏ đến lớn.
- Trường hợp khách đặt bàn theo ID nhưng bàn đó đã thuộc về một vị khách khác, tuy nhiên nhà hàng vẫn còn bàn trống. Thì nhân viên sẽ ưu tiên chọn bàn cho khách với ID lớn hơn ID hiện tại trước. Sau đó nếu ID không thể tăng được nữa (ID = MAXTABLE) thì sẽ quay lại xem xét tiếp từ bàn đầu tiên (ID = 1).
  - Ví dụ nhà hàng có 5 bàn được đánh số từ 1 đến 5 và khách đặt bàn với ID = 3, nhưng bàn này đã có khách đặt. Thì nhân viên sẽ ưu tiên chọn bàn cho khách theo thứ tự: $4 \longrightarrow 5 \longrightarrow 1 \longrightarrow 2$.
- Trường hợp nếu như khách hàng đến nhà hàng nhưng nhà hàng đã kín bàn thì khách hàng sẽ được nhân viên xếp vào hàng chờ. Số lượng khách hàng tối đa trong hàng chờ bằng với số lượng bàn tối đa trong nhà hàng. Khách hàng (tới (đặt bàn với ID hoặc không) trong hàng chờ sẽ được vào nhà hàng dùng bữa ngay khi có bàn trống theo thứ tự (First In First Out). Đồng nghĩa với việc thông tin của bàn cũng sẽ được cập nhật theo thông tin của khách vừa vào. Tuy nhiên, vị trí trong hàng chờ có thể thay đổi khi có yêu cầu từ chủ nhà hàng (xem lệnh **SQ**).

REGM <NAME> <AGE> <NUM> - O(N):
Lệnh này dùng để gộp bàn và đặt bàn cho một nhóm khách VIP.

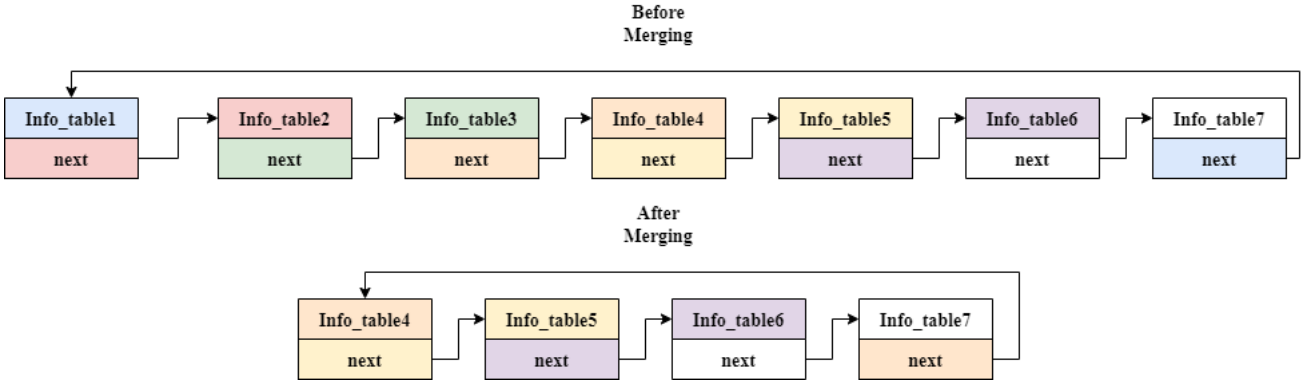
Giả sử một bàn tại nhà hàng chỉ có thể phục vụ cho một người. Thì khi có các vị khách VIP đến nhà hàng và họ đi theo nhóm, chủ nhà hàng bắt buộc phải gộp bàn lại để chiêu đãi các vị khách này. Ngoài ra, mặc dù đi theo nhóm nhưng nhóm khách VIP này vẫn được xem như một vị khách với NAME và AGE của vị khách ứng với lệnh này.

Việc gộp bàn được tiến hành như sau:

- Khi có lệnh từ chủ nhà hàng, nhân viên sẽ tiến hành gộp **các bàn trống được xếp liên tục nhau**, với số bàn cần gộp là NUM. Và việc gộp bàn chỉ diễn ra một lần tại một

khoảng thời gian. Nghĩa là trong nhà hàng nếu đang có bàn gộp thì khi nhận được một lệnh gộp bàn khác thì lệnh gộp bàn đến sau sẽ bị bỏ qua.

- Nếu có nhiều dãy bàn trống liên tiếp nhau thì nhân viên sẽ ưu tiên chọn dãy bàn trống bắt đầu từ bàn có ID lớn nhất và đây cũng chính là ID của bàn gộp kèm theo NAME và AGE của vị khách đại diện nhóm khách VIP đó.
  - Ví dụ nhà hàng có 7 bàn được đánh số từ 1 đến 7 và bàn 5 đã được đặt. Khi có lệnh gộp bàn với NUM = 4. Thì nhân viên sẽ có 3 lựa chọn để gộp bàn trống như sau:
    1. $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 4$.
    2. $6 \longrightarrow 7 \longrightarrow 1 \longrightarrow 2$.
    3. $7 \longrightarrow 1 \longrightarrow 2 \longrightarrow 3$.
  - Tuy nhiên, theo yêu cầu đề ra thì nhân viên sẽ chọn cách gộp bàn ở **lựa chọn số 3**.



Hình 2: Minh họa danh sách bàn trước và sau khi tiến hành gộp bàn.

- Ngoài ra, tại thời điểm khi nhận được lệnh gộp bàn, nếu như không tìm được các dãy bàn liên tiếp để gộp hoặc nhà hàng kín bàn thì các vị khách này sẽ bỏ đi. Lệnh gộp bàn sẽ bị bỏ qua.

CLE <ID> - O(N):

- Đối với bàn đơn: Lệnh này sẽ xóa thông tin của khách hàng tại bàn ứng với ID và chuyển thành bàn trống. Nghĩa là khi khách hàng dùng bữa xong, thanh toán và ra về, thì trạng thái của bàn đó sẽ là bàn trống và nhân viên có thể sắp xếp cho các vị khách kế tiếp. Trường hợp nếu trạng thái của bàn đó đã trống hoặc ID của bàn không tồn tại trong danh sách bàn thì không cần làm gì.
- Đối với bàn gộp: đầu tiên, nhân viên sẽ xóa thông tin khách tại bàn gộp ứng với ID đó. Tiếp theo, khôi phục lại vị trí của các bàn trở lại như cũ (như trước khi gộp). Chuyển đổi danh sách bàn từ After Merging sang Before Merging (đảo ngược chiều chuyển đổi ở hình 2)

PS [NUM] - O(N):

Lệnh này được dùng để in ra tên của của NUM vị khách đang ở nhà hàng và đến nhà hàng gần nhất (bao gồm cả khách đã có bàn và khách trong hàng chờ). Bắt đầu in từ khách vừa đến quán gần nhất trở về trước, với giá trị của NUM thuộc đoạn từ 1 đến 2*MAXTABLE. Trong đó:

1. Nếu NUM $\leq$ tổng số lượng khách đang có, in ra chuỗi kết quả ứng với từng khách theo định dạng: "NAME/n", với NAME là tên của khách.
2. Nếu tất cả các bàn trong nhà hàng đều trống, thì lệnh này sẽ in chuỗi "Empty/n".
3. Nếu NUM $>$ tổng số lượng khách đang có nhưng vẫn $\leq$ 2*MAXTABLE, thì sẽ in theo định dạng tương tự trường hợp 1, nhưng cho toàn bộ khách hàng đang có mặt ở nhà hàng.

PQ [NUM] - O(N):

Lệnh này được dùng để in ra tên của của NUM vị khách, bắt đầu từ khách đến sớm nhất trong hàng chờ trở về sau, với giá trị của NUM thuộc đoạn từ 1 đến MAXTABLE. Trong đó:

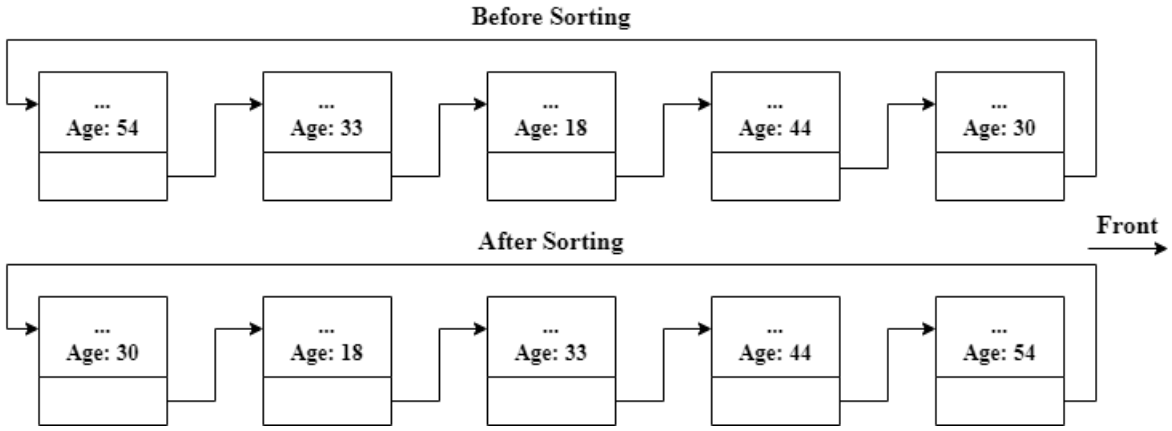
1. Nếu NUM $\leq$ số lượng khách đang trong hàng chờ, in ra chuỗi kết quả ứng với từng khách theo định dạng: "NAME/n", với NAME là tên của khách.
2. Nếu nhà hàng vẫn còn bàn (hàng chờ trống) thì lệnh này sẽ in chuỗi "Empty/n".
3. Nếu NUM $>$ số lượng khách đang trong hàng chờ nhưng vẫn $\leq$ MAXTABLE, thì sẽ in theo định dạng tương tự trường hợp 1, nhưng cho toàn bộ khách trong hàng chờ.

SQ <NUM> - O(N^2):

Khi nhận được lệnh trên, nhân viên sẽ sắp xếp lại hàng đợi (theo giải thuật Selection Sort) nhưng chỉ ưu tiên cho NUM người lớn tuổi nhất, với giá trị của NUM thuộc đoạn từ 1 đến MAXTABLE. Nếu có nhiều người có cùng tuổi trong hàng chờ thì người nào đến trước thì được ưu tiên xếp trước. Sau khi đã sắp xếp lại hàng chờ, in tên của các vị khách trong hàng chờ theo định dạng: "NAME/n", với NAME là tên của từng vị khách. Trong đó:

1. Nếu NUM $\leq$ số lượng khách đang trong hàng chờ, thì sắp xếp lại hàng chờ, ưu tiên cho NUM vị khách lớn tuổi nhất.
2. Nếu hàng chờ trống thì lệnh này sẽ in chuỗi "Empty/n".
3. Nếu NUM $>$ số lượng khách đang trong hàng chờ nhưng vẫn $\leq$ MAXTABLE, thì sắp xếp lại hàng chờ, ưu tiên theo độ tuổi từ lớn đến nhỏ cho toàn bộ khách trong hàng chờ.

Ghi chú: Trong bài tập lớn này, giả định không có khách hàng nào vừa trùng tên và vừa trùng



Hình 3: Hàng chờ trước và sau khi tiến hành sắp xếp với NUM = 3.

tuổi xuất hiện cùng lúc tại nhà hàng. Ngoài ra, mỗi một bàn trong nhà hàng chỉ có thể phục vụ cho một khách duy nhất.

2.3 Yêu cầu

Để hoàn thành bài tập lớn này, các em phải:

- Tải xuống tập tin initial.zip và giải nén nó.
- Sau khi giải nén sẽ được 3 files: main.cpp, main.h, restaurant.cpp. Các em **KHÔNG ĐƯỢC** sửa đổi các file main.cpp, main.h.
- Sinh viên được quyền chỉnh sửa bất kỳ nội dung trong file restaurant.cpp để hiện thực bài toán. Tuy nhiên, không được thay đổi prototype của hàm simulate.
- Đảm bảo rằng chỉ có một lệnh **include** trong file **include** trong file restaurant.cpp đó là **♯include "main.h"**. Ngoài ra, không cho phép có một **include** nào khác trong file này.
- Không được sử dụng cấu trúc dữ liệu array trong bài làm. Hệ thống chấm bài sẽ bỏ qua các bài làm có chứa các từ khóa hoặc các toán tử liên quan đến array. Do đó, bài làm sẽ không được chấm.
- Môi trường dùng để chấm bài là g++ (MinGW-W64 i686-ucrt-posix-dwarf) 11.2.0

3 Nộp bài

Các em chỉ nộp file: **restaurant.cpp**, trước thời hạn được đưa ra trong đường dẫn "Assignment 1 Submission". Có một số testcase đơn giản được sử dụng để kiểm tra bài làm của các em nhằm đảm bảo rằng kết quả của em có thể biên dịch và chạy được. Các em có thể nộp bài bao nhiêu lần tùy ý nhưng chỉ có bài nộp cuối cùng được tính điểm. Vì hệ thống không thể chịu tải khi

quá nhiều em nộp bài cùng một lúc, vì vậy em nên nộp bài càng sớm càng tốt. Các em sẽ tự chịu rủi ro nếu nộp bài sát hạn chót. Khi quá thời hạn nộp bài, hệ thống sẽ đóng nên các em sẽ không thể nộp nữa. Bài nộp qua email không được chấp nhận.

4 Harmony

Trong đề thi có thể có các câu hỏi liên quan đến nội dung bài tập lớn (phần mô tả câu hỏi sẽ được nêu rõ là dùng để harmony cho bài tập lớn, nếu có) . Điểm của các câu hỏi harmony sẽ được scale về thang 10 và sẽ được dùng để tính lại điểm của các bài tập lớn. Cụ thể:

- Gọi x là điểm bài tập lớn.
- Gọi y là điểm của các câu hỏi harmony sau khi scale về thang 10.

Điểm cuối cùng của bài tập lớn sẽ được tính theo công thức trung bình điều hòa sau:

$$\mathbf{Assignment\_Score = 2 * x * y \ / \ (x + y)}$$

5 Xử lý gian lận

Bài tập lớn phải được sinh viên TỰ LÀM. Sinh viên sẽ bị coi là gian lận nếu:

- Có sự giống nhau bất thường giữa mã nguồn của các bài nộp. Trong trường hợp này, TẤT CẢ các bài nộp đều bị coi là gian lận. Do vậy sinh viên phải bảo vệ mã nguồn bài tập lớn của mình.
- Sinh viên không hiểu mã nguồn do chính mình viết, trừ những phần mã được cung cấp sẵn trong chương trình khởi tạo. Sinh viên có thể tham khảo từ bất kỳ nguồn tài liệu nào, tuy nhiên phải đảm bảo rằng mình hiểu rõ ý nghĩa của tất cả những dòng lệnh mà mình viết. Trong trường hợp không hiểu rõ mã nguồn của nơi mình tham khảo, sinh viên được đặc biệt cảnh báo là KHÔNG ĐƯỢC sử dụng mã nguồn này; thay vào đó nên sử dụng những gì đã được học để viết chương trình.
- Nộp nhầm bài của sinh viên khác trên tài khoản cá nhân của mình.

Trong trường hợp bị kết luận là gian lận, sinh viên sẽ bị xử lý theo kết luận của Hội đồng giảng viên giảng dạy môn học này.

KHÔNG CHẤP NHẬN BẤT KỲ GIẢI THÍCH NÀO VÀ KHÔNG CÓ BẤT KỲ NGOẠI LỆ NÀO!

——————————HẾT——————————