Bài toán: Xây dựng ứng dụng quản lý order trong cửa hàng

Mô tả
Hàng ngày, người quản lý cửa hàng cần quản lý danh sách các order của khách hàng đã được đặt, hủy, hoàn thành. Cuối ngày, thông tin các order này sẽ được lưu ra file text để tra cứu về sau nếu cần. Thông tin 1 món ăn khi đặt sẽ gồm: • Mã món ăn/Tên món ăn. • Số lượng đĩa đặt(mỗi món ăn đặt 1 lần = 1 đĩa, đặt nhiều lần → nhiều đĩa). • Số lượng đĩa trả (sau khi bếp làm xong sẽ trả cho khách). • Ghi chú Thông tin order cần quản lý gồm • Thời gian tạo order. • Mã nhân viên/tên nhân viên ghi order. • Mã bàn/số thứ tự bàn. • Danh sách món ăn đã đặt • Tổng số món đặt (tính dựa trên yêu cầu của khách). • Tổng số đĩa đặt. • Tổng số món trả (sau khi bếp làm xong sẽ trả cho khách). • Tổng số đĩa đã trả. • Thời gian cập nhật đơn hàng lần cuối (mỗi lần trả đơn sẽ cập nhật vào thời gian này). • Trạng thái đơn hàng (đang phục vụ, đã thanh toán, đơn hủy) Khi lên đơn, mặc định trạng thái đơn hàng là đang phục vụ. Chỉ có thể hủy đơn hàng nếu đơn đó chưa trả món nào (nhà bếp chưa làm), ngược lại sẽ không được phép hủy. Khách có thể hủy món (nếu nhà bếp chưa làm), khi đó số lượng đĩa trả sẽ là 0, và có ghi chú là Khách hủy, bếp hủy,... Một bàn có thể order nhiều lần, có thể hủy order và có thể có nhiều order cùng mã bàn. Khi khách thanh toán, hệ thống sẽ duyệt và tổng hợp hết những order của khách đã được thực hiện và tính tiền dựa trên tổng số lượng món/đĩa mà bếp đã làm và trả cho khách. Để dễ quản lý các order này, chúng ta sẽ dùng một cấu trúc danh sách liên kết đơn để xử lý. Thông tin một phần tử của cấu trúc được khai báo như sau

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_NAME 50
#define MAX_NOTE 100

// Trạng thái đơn hàng
typedef enum {
    DANG_PHUC_VU,
    DA_THANH_TOAN,
    DON_HUY
} TrangThai;
```

```c
// Cấu trúc món ăn
typedef struct Dish {
    char maMon[MAX_NAME];
    char thoiGian[20];
    int soLuongDat;
    int soLuongTra;
    char thoiGianCapNhat[20];
    char ghiChu[MAX_NOTE];
    struct Dish* next;
} Dish;

// Cấu trúc đơn hàng
typedef struct Order {
    char thoiGian[20];
    char tenNhanVien[MAX_NAME];
    int maBan;
    Dish* danhSachMon;
    int tongSoMon;
    int tongSoDiaDat;
    int tongSoMonTra;
    int tongSoDiaTra;
    char thoiGianCapNhat[20];
    TrangThai trangThai;
    struct Order* next;
} Order;
```

Đầu vào sẽ là dạng

search_order <mã bàn> Trả về con trỏ tới order hiện có của bàn có <mã bàn> theo đúng thứ tự. Nếu <mã bàn> chưa tạo order hoặc đã hoàn thành thì in ra "Chua co order". VD. `Order*` **search_order (int table_id)**
print_order <mã bàn> In ra chi tiết các order của mã bàn nếu mà bàn đó đang được phục vụ (in ra các món đặt và món đã trả). Nếu bàn đó chưa tạo order hoặc đã hoàn thành thì in ra thông báo.
create_order <mã bàn> <mã nhân viên> <thời gian tạo YYYY-MM-DD HH:MM:SS> Tạo ra 1 bàn mới khi chưa có bàn hoặc bạn đã thanh toán xong.
add_dish <mã bàn> <mã món> <số lượng> <ghi chú> Gọi món cho bàn có mã **<mã bàn>.** Nếu bàn đó đã có trong danh sách thì thêm món vào tiếp danh sách các món đang gọi, ngược lại sẽ tạo order mới cho bàn đó (bàn chưa order hoặc đã thanh toán sẽ được tạo order mới). Thời gian order sẽ lấy là thời gian lúc gọi hàm. Hàm trả về 1 nếu thành công và 0 nếu thất bại. VD. int **add_dish(int table_id, char *** `maMon, int quantity, char* notes)`
update_dish <mã bàn> <mã món> <số lượng> Hàm trả món cho bàn <mã bàn>, và cập nhật vào `thoiGianCapNhat` Hàm trả về 1 nếu thành công và 0 nếu thất bại. VD. int **update_dish(int table_id, char *** `maMon, int quantity)`
cancel_dish <mã bàn> <mã món> <ghi chú> Hàm hủy món, hàm trả về 1 nếu thành công và 0 nếu thất bại. Ghi nhận vào thời gian cập nhật. VD. int **cancel_dish(int table_id, char *** `maMon, char* notes)`

cancel_order <mã bàn> Hàm hủy order ứng với <mã bàn>, hàm trả về 1 nếu thành công và 0 nếu thất bại. VD. int **cancel_order(int table_id)**
create_bill <mã bàn> Hàm thanh toán, in ra danh sách các order đã thực hiện của bản <mã bàn>. Sau khi thanh toán, thông tin sẽ được lưu vào file văn bản. VD. void **create_bill(int table_id)**

Ví dụ sử dụng

? **add_dish** NV012 01 “Bun ngan” 2 “” NV012 01 “Bun bo” 1 “chin tai” #
Giải thích Nhân viên NV012 ghi order của bàn 01. Bun ngan : số lượng 2. Bun bo: số lượng 1 và yêu cầu chín tái. Nếu bàn 01 này chưa có order thì tạo thêm order mới, ngược lại thì thêm vào order đang có.

? **create_order** NV012 01 2025-03-29 12:15:07 #
Giải thích Nhân viên NV012 tạo order của mới cho bàn 01 với thời gian tạo 2025-03-29 12:15:07

? **update_dish** NV007 01 “Bun ngan” 2 NV007 01 “Bun bo” 1 #
Giải thích Nhân viên NV007 trả món cho bàn 01.

? **cancel_dish** NV007 01 “Bun ngan” “đổi món” #
Giải thích Nhân viên NV007 yêu cầu hủy món Bun ngan với yêu cầu của khách là đổi món

? **cancel_order** NV007 02 #
Giải thích

Nhân viên NV007 yêu cầu hủy order bàn 02

? **create_bill** NV007 02 #
Giải thích Nhân viên NV007 yêu cầu tạo bill cho bàn 02, chỉ in ra danh sách các món ăn đã trả món. Ghi ra file chi tiết lịch sử order của bàn 02.

Ví dụ đầu vào và đầu ra

INPUT
? **create_order** NV012 01 2025-03-29 12:15:07 NV012 02 2025-03-29 12:15:07 # ? **add_dish** NV012 01 “Bun ngan” 2 “” NV012 01 “Bun bo” 1 “chin tai” NV012 02 “Bun ngan” 2 “” NV012 01 “Mi xao” 1 “” # ? **update_dish** NV007 01 “Bun ngan” 2 NV007 01 “Bun bo” 1 # ? **cancel_dish** NV007 01 “Bun ngan” “đổi món” # ? **print_order** 02 ? **create_bill** NV007 01 #
OUTPUT
2025-03-31 12:10:15 Ban 02 Bun ngan : 02 Tong so mon : 1 Tong so dia: 2 2025-03-31 12:15:15 Ban 01 Bun ngan : 02 Bun bo : 01 Tong so mon : 2 Tong so dia : 3