Programming Assignment 2 - Part II

Implementation of a Service Queue ADT
DUE: Monday Feb 29 by 11:59PM

You go to a popular restaurant that doesn't take reservations. When you
get there, you just have to wait in line for a table. These days, most
restaurants will give you some kind of buzzer and will signal you when your
table is ready.

The restaurant essentially has to manage a queue of buzzers.
Implementation of an ADT supporting this management is what you will be
doing in this assignment.

The ADT interface is specified in the provided file **sq.h**. The first thing
it does is (partially) specify a type **SQ** for a service queue. A client
program uses service queue pointers (type **SQ ***).

Below is the **SQ typedef** and function prototypes as specified in **sq.h**

```
// incomplete type:  struct service fields specified in
//     implementation file
typedef struct service_queue SQ;

extern SQ * sq_create();

void sq_free(SQ *q);

extern void sq_display(SQ *q);

extern int  sq_length(SQ *q);                // O(1)

extern int  sq_give_buzzer(SQ *q);           // O(1) amortized

extern int sq_seat(SQ *q);                   // O(1)

extern int sq_kick_out(SQ *q, int buzzer);   // O(1)

extern int sq_take_bribe(SQ *q, int buzzer); // O(1)
```

Each function has a banner comment above it specifying the required behavior **AND** a required runtime.

Your main job: write an implementation file `sq.c` meeting *all* of the requirements specified in `sq.h`

STEP 1: read the banner comments in sq.h and make sure you understand the required behavior of all of the functions.

STEP 2: For reference, you have been given a complete implementation file `sq_slow.c`. The catch is: it doesn't meet all of the runtime requirements.

Read through this implementation.

Reading through this pretty simple implementation should help you understand the expected behavior (but not how to meet the runtime requirements). This implementation uses the previously existing ADT for lists.

You will also notice an application program `driver.c`. It initializes a service queue and starts a simple interactive interface that lets the user perform the various operations (basically a one-to-one correspondence with the functions plus a quit command).

STEP 3: do a runtime analysis for each of the operations as implemented in `sq_slow.c`. This will be part of a writeup you will submit as part of the assignment.

STEP 4: Now the fun part! Start designing *your* implementation of the service queue which meets the runtime requirements. Advice:

- Take a "blank slate" approach: do NOT try to modify sq_slow.c to meet the runtime requirements.
- Start with pencil and paper.
- Remember, you get to specify what goes in the **service_queue** structure.
- Take advantage of the fact that the buzzer-IDs are not just any old integers -- they start from zero and increase from there.

- It will probably **NOT** be useful to use the given **list** ADT like **sq_slow.c** does. This does not mean that **some kind of** linked list implementation won't be useful. Take a blank slate approach to this aspect as well.
- Note that almost all of the runtime requirements are O(1). So if you find yourself writing a loop, think twice. (In the case of **sq_give_buzzer**, you probably will end up with a loop, but read the runtime requirements carefully).

When you think you have a correct "design" (i.e., an organization of the data which enables the specified run times), feel free to post a Piazza note to instructors only for a sanity check. Such posts are expected to be very clear if you expect a meaningful response!!!

STEP 5: Implement, debug and test your implementation.

Deliverables

You will submit a single zip file containing the following:

- All source code:
    - sq.c
    - sq.h (unmodified)
    - driver.c
    - any other source files you wrote for your solution. You probably won't have any other files, but if you decide to partition your code, it is acceptable.
- Makefile:
    the makefile must enable the following:
        > make sq.o
        > make fdriver

    The target fdriver is just the given driver program which uses your (fast) implementation of the ADT.

    Just modify the given makefile…

- writeup.pdf:
    Your writeup must include the following:

A. Runtime analysis of all of the functions as implemented in `sq_slow.c`
B. A description of your design -- i.e., how the data is organized and how it enables the runtime requirement of each operation.

You will probably create your writeup using MS Word, Google Docs or something like that. However, if it is very, very neat, a hand-written submission is ok. The scan *must* also be of high quality.

We do not want you to write a book! Keep it simple and to-the-point. You will almost certainly want to include diagrams to aid in explaining your solution.

Submission

Your zip file will be uploaded to Blackboard as usual. Note that there will be separate submissions for Part-I and Part-II.