Usman Siddiqui

Usiddi4@uic.edu

Part 1:

Runtime Analysis of sq\_slow.c

Runtime of sq\_free: O(N\_b), N\_b number of buzzers

Runtime of sq\_display: O(N)

Runtime of sq\_length: O(1)

Runtime of sq\_give\_buzzer: O(1)

Runtime of sq\_seat: O(1)

Runtime of sq\_kick\_out: O(N)

Runtime of sq\_take\_bribe: O(N)

Part 2:

To make sure I get correct runtime requirements I used an implementation with an array and the use of 2 linked lists. Bribes keeps track of bribes that were done as they are an oddity that messes up the program, and every bribe changes the buzzer\_id setting to 2. Buzzer\_ID settings are set to 1 as in the list, 0 not in the list, or 2 bribe. This setting I refer to is the value of the array at location of the buzzer\_ID. When printing the queue, I make it so I first print out all the bribed buzzers first then go on to print out the others. The display still meets O(N) since there are never more than N bribes, and the displaying skips to the first buzzer\_ID. Thus the display is set to O(N) since the number of loops < 2N, N being the number of buzzer\_ID’s. There are also parts of the queue used in different ways: size, length, and number. The size is the size of the array, the length is the actual length of the array (as in how much of it has useful data or how much of the array has been used so far), and number is the number of buzzer\_ID’s. the sq\_give\_buzzer function will sometimes get an O(M) runtime whenever the array is full, but otherwise it will give a O(1) runtime, and since the array doubles it should average to O(1). The other functions should be O(1) since they mainly accessing or writing to an array.