Saqib & Rashan

Description:

Our project is created for purposes of a restaurant setting with a continuous flow of customers coming in. The restaurant has a set number of two seat and four seat tables, and the program minimizes wait time for incoming customers. For purposes of this project, the number of two seat tables is set to four, and the number for four seat tables is set to 7. Two seat tables are reserved solely for groups of one and two, and four seat table are reserved for groups of 3-24 people. Also, for testing purposes, the average time that a single group takes to eat is set to 15 seconds. After 15 seconds, the group leaves the table. We have a waiting list for the event that all tables are full, and the program continuously checks for open seat to move people out of the waiting list and seats them. Also, if the wait time for a group in the waiting list exceeds a certain amount (25 seconds), the group leaves. Since the restaurant does not have a table size greater than 4 seats, if a group larger than 4 comes in (and seats are not available), the program waits until the correct number of tables are cleared up and the group is seated together. Groups larger than 24 cannot be accommodated, and are asked to reserve elsewhere. The goal is to maintain seats open for smaller groups alongside larger groups. To keep track of groups, each group is assigned a group name based on user input.

Data structures used:

Array for tables

Deques for waiting list

Struct for groups

To begin with, we used a Struct to create groups of customers coming in. The struct allows us to easily associate pertinent information with every group that walks in. For example, we need to keep track of how many people are in a single group, the time they entered, the name of the group, and, for purposes of keeping track of table space, a bool occupied. The bool occupied is used in conjunction with the array used to represent tables.

We used arrays of type group to represent the tables. The arrays were the best option for a couple reasons: 1.) the number of tables is not changing. Arrays are the most efficient static data structure. 2.) The array allows us to access any index (corresponding to a table). The way we keep track of whether or not a table is in use or not is by using the bool occupied value associated with the groups. Initially all indices of a table are set to false. When a group sits down, the value of the table changes to match the bool value of the group (in this case “true” to indicate table is in use).

Lastly, we used a deque for keeping track of groups on the waiting list. We needed a deque because it allows us to add or delete on either end. The way the waiting list is set up is first person in line is the first person to be seated. Also, if a group is added to the waiting list and the wait time is greater than 25 seconds, the group leaves. Hence the need for deleting data from both ends.

For representing the tables, we used an array. We used an array because, the number of tables is statically allocated. An array also allows us to access individual indices to seat and unseat people (seated table is indicated by 1. Empty table is 0).

How to use the code:

*this test case is time sensitive.

To test functionality of seating 1-2 people

1. Enter number of people in group (1 or 2)
2. Enter group name
3. Press enter if you would like to add more customers coming in
4. There are four two seat tables. If you create 4 groups successively (within 15 seconds), the groups will be added to the waiting list until seats open up

To test functionality of seating 3-4 people

1. Enter number of people in group (3 or 4)
2. Enter group name
3. Press enter if you would like to add more customers coming in
4. There are seven four seat tables. If you create 7 groups successively (within 15 seconds), the groups will be added to the waiting list until seats open up

To test groups larger than 4, less than 25

1. Enter large group number at any time when prompted to enter a group.
2. Large group will be added to waiting list until seats open up.

Test case:

1. enter four groups of 1 or 2 successively (with group number followed by name)
    a. outcome will be all tables are full
2. enter one more group of 1 or 2 followed by name
    a. waiting list will be increased
3. Wait 15 seconds. Add one more group of 1-2. Group will be shown as added to the first position as table is cleared up.
4. To test groups of 3-4, repeat steps 1-3. Tables will be shown as full when 7 groups are added within 15 seconds. Waiting list works in same manner.
5. Groups larger than 4 are seated in combined 4 seat tables.
    a. Create group of 20. Enter a name for group
    b. When tables are available, you should see 5 of the 4 seat tables become 1. This indicates that the group of 20 was seated together.

Partner Documentation: Saqib and Rashan

*****Entire project was worked on together at the same time together. Specific documentation provided for formality.

- Both commented on code
- Saqib – Wrote documentation
- Rashan- wrote function to seat groups (filltable function).
- Rashan – wrote wait time remaining function.
- Saqib- Wrote check time function (checks when table needs to be cleared.)
- Saqib-Wrote twentyseatFillcheck function (used to check availability of seat for large groups)
- Main function was written as a combined effort.