**ChauncyKent\_01\_01 Technical Design Document**

**Name:** Chauncy Kent

**Date Created:** 8/23/2025

**Program Description:**

The program starts with a set number of tickets available for presale. Using a loop, the program asks each customer in turn how many tickets they would like to purchase, making clear the limits (max of 4, or less if total remaining tickets are less than 4). Once the transaction is complete, the program updates the total ticket amount and adds to the customer count. Finally, when all the tickets have been sold, the program terminates the loop and displays the total number of customers there were served.

**Functions used in the Program (list in order as they are called):**

1. **Function Name:** main

**Description:** The main function where initial variable values are established

    and helper functions are called.

**Parameters:** None.

**Variables:** tickets(integer) – An integer used to track the remaining number of tickets.

customers(integer) – An integer used as an accumulator to track the number of customers served.

**Logical Steps:** 1. Initializes variables with default values.

2. Initializes a loop to cycle through customers until no tickets remain.

3. (Per loop) subtracts the number of tickets purchased from the total remaining.

4. (Per loop) add 1 to the number of customers served.

5. Display the total number of customers served.

**Returns:** None.

2. **Function Name:** buy\_ticks

**Description:** Takes in the number of tickets as an argument and returns the number of tickets purchased by the customer.

**Parameters:** tickets(integer) – A number representing the number of remaining tickets.

**Variables:** message(string) – The message to display to the customer.

max\_ticks(integer) – A number representing the maximum number of tickets the customer may purchase.

order(integer) – A number representing the number of tickets the customer purchased.

**Logical Steps:** 1. Evaluates remaining tickets and sets an appropriate message.

2. Evaluates remaining tickets and sets an appropriate maximum number of tickets that the customer may purchase.

3. Initializes an input validation loop to take in and validate customer input

4. Assigns customer input to the order variable

5. Attempts to convert the input to an int. If this fails, the loop restarts.

6. Checks to see if the input is within the bounds. If not, the loop restarts.

7. Returns the value of the customer's order.

**Returns:** The order variable, an integer representing the number of tickets purchased by the customer.

3. **Function Name:** set\_max\_ticks

**Description:** Takes in the number of tickets and a default maximum number of tickets as arguments and returns the appropriate maximum number of tickets the next customer may buy.

**Parameters:** tickets(integer) - A number representing the number of remaining tickets.

maximum(integer) – A number representing the maximum number of tickets a customer may purchase. Defaults to 4.

**Variables:** max\_tickets(integer) – A number representing the maximum number of tickets a customer may purchase.

**Logical Steps:** 1. Initializes the max\_tickets variable and assigns it the default value specified by the maximum argument.

2. Evaluates if fewer tickets are remaining than the current maximum, and if so, adjusts the maximum to match the current remaining number of tickets.

3. Returns the appropriate maximum tickets the next customer may purchase.

**Returns:** The max\_tickets variable, an integer that represents the maximum number of tickets the next customer may purchase.

4. **Function Name:** set\_message

**Description:** Takes in the number of tickets and the max number of tickets as arguments and returns the appropriate message for the next customer.

**Parameters:** tickets(integer) – A number representing the number of remaining tickets.

max\_ticks(integer) – A number representing the maximum number of tickets a customer may purchase.

**Variables:** message(string) – The message to display to the customer.

**Logical Steps:** 1. Initializes the message variable with an empty string.

2. Evaluates if the number of remaining tickets is greater than or equal to the maximum number of tickets.

3. Assigns the appropriate message to the message variable based on the results of the previous evaluation.

4. Returns the appropriate message to display to the next customer.

**Returns:** The message variable, a string to be used as the message to the next customer.

**Logical Steps:**

1. main is called first by the user to kick off the program.
2. main calls buy\_ticks to start the ticket buying process.
3. buy\_ticks calls set\_max\_ticks to determine the current appropriate maximum to allow the next customer to purchase.
4. buy\_ticks calls set\_message to determine the most appropriate message to display to the next customer.

**Link to your repository:** <https://github.com/ChauncyJKent/COP2373>

**Output Screenshot:**

