TEAM – Bad Idea (P3)

GITHUB LINK- https://github.com/shraddha0329/Capstone_Queue-Manager

PROJECT DESCRIPTION - This project about the entry queue manager is helpful in efficiently managing the entry of viewers in a stadium with 'N' number of entry gates. The main goal of this Project is to minimize the wait time and the crowding of viewers.

FEATURES:

1.MULTIPLE GATES: stadium has 'N' number of gates, each gate will act as an entry queue.

2.USAGE OF DYNAMIC QUEUE: Dynamic queue is used for switching the queue in order to get quick entry.

3.QUEUE MONITORING: Length of each queue is monitored which helps in estimating the wait time of a person standing in the queue.

4.GUIDANCE: We will inform the viewer if there is a change in queue and the time estimated for their entry.

5.SPECIAL ENTRY: We will provide special entry to VIP's and the disabled.

PSEUDOCODE:

Create a Random class:

Initialize a seed value 's' to 123456789

Create a Node structure to represent each person in the queue:

Define a data field to hold the person's identifier

Define a next field as a reference to the next node in the queue

Create a Queue class:

Initialize head and rear pointers to null

Method enqueue(value):

Create a new node with the given value

If the queue is empty:

Set both head and rear to point to the new node

Otherwise:

Point the new node to the rear and update rear to point to it

Method dequeue():

If the queue is empty:

Print "Queue is empty" and return -1

Otherwise:

Get back the data from the head node

Move the head pointer to the next node

Delete the original head node

Return the data that got back

Method isEmpty():

Return true if the head pointer is null, indicating an empty queue

Method size():

Travel across the queue from head to rear, counting the number of nodes

Return the count

Method calculateTime(p):

Multiply the number of people in the queue by the time it takes to pass through the gate (p)

Return the result

Method getLastPerson():

If the queue is empty:

Return -1

Otherwise:

Return the data of the last person in the queue

Method moveLastPersonTo(otherQueue):

If the queue is not empty:

Dequeue the last person from this queue

Enqueue the last person into the provided 'otherQueue'

Create a function allQueuesEqual(queues, N) to check if all queues have equal sizes:

Get the size of the first queue

Iterate through the remaining queues:

If any queue's size is different from the first queue's size:

Return false

Return true

In the main function:

Create an instance of the Random class

Take the input form the user to enter the total number of persons (M)

Take the input form the user to enter the total number of queues (N)

Take the input form the user to enter the time it takes to pass through the gate (p)

Create an array of 'N' Queue objects

Distribute M/2 persons randomly among the N queues:

For each person up to M/2:

Generate a random queue number

Enqueue the person into the randomly selected queue

Output the time for each queue to pass through the gate:

For each queue:

Calculate and print the time taken by that queue to pass through the gate

Initialize switchCount to 0 and maxIterations to 10000

Repeat until all queues have equal sizes or maximum iterations reached:

Find the queue with the longest waiting time:

Iterate through each queue, calculating the time each queue takes to pass through the gate

Identify the queue with the maximum time

Find the queue with the shortest waiting time:

Iterate through each queue, calculating the time each queue takes to pass through the gate

Identify the queue with the minimum time

Move the last person from the queue with the longest waiting time to the queue with the shortest waiting time

Increment switchCount by 1

Output the switch operation performed

If maximum iterations reached:

Output "Maximum number of iterations reached. Convergence not achieved."

Else:

Output the time for each queue to pass through the gate after switching:

For each queue:

Calculate and print the time taken by that queue to pass through the gate

Output the total number of switches made

WHY QUEUE? :

1.FIFO(first in first out):This feature of queue makes it very convenient for us to make an entry. As the items are inserted at the end of the queue and removed from the front. The first item to be inserted is the first to be removed.

2.It maintains the order in which elements are inserted

3.Enqueue: It's a key part of using a queue, which is like a line you stand in. When you enqueue something, you're adding it to the end of the line. It's like joining the back of the queue.

4.Dequeue: Dequeuing is another essential operation in a queue, and it's like taking something out of the line. When you dequeue, you're removing the first item that was added to the queue.

Queues lies in their ability to manage elements in a way that ensures they are processed in the order they were added, making them valuable in scenarios

requiring ordered, predictable processing. These qualities of queue make it the most convenient data structure that could be used in our entry queue manager.