LAB SESSION 3: QUEUE DATA STRUCTURE

AIM: To implement a task scheduler using Queue Data Structure.

PROBLEM DEFINITION: Create a structure TASK with the following data fields:
TaskId(int)
TaskTitle (char [])
TaskDuration (int) to be interpreted in seconds
Status: Idle/Queued/Completed

Create an array of TASK comprising of 10 tasks. (Preferably Take input from file or initialise at compile time)

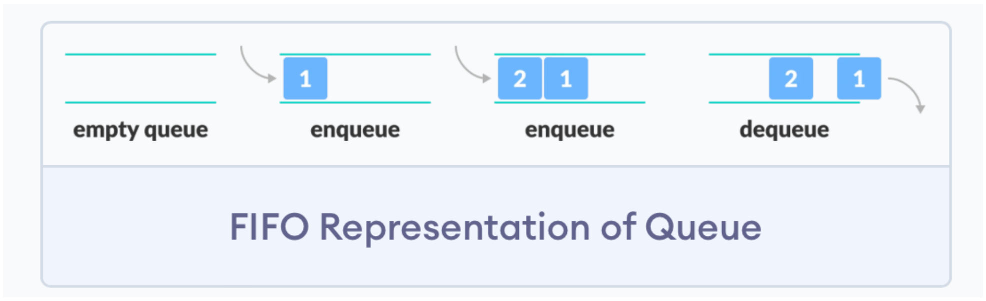
Implement a Queue data structure to schedule tasks from the above array. Max size of Queue is 5. Your options should include the following:
1. Enter the task id of the tasks to be scheduled (insert these in a queue). If status is queued or completed then do not allow scheduling. If the queue is full, ask the user to wait for a certain amount of time (Print the minimum time of waiting and the max waiting time to the user)

2. Run the Tasks: delete the task at start of queue and run it for the given taskDuration. (Use delay() function to suspend the running of the program for the indicated time). Once the task is completed prompt the user the menu and continue based on his choice.

3. Display the details of tasks that are queued up.

THEORY: A queue is a data structure that follows the FIFO (first in first out) principle. In this data structure we have a Head and a tail pointer which refers to the element at start and end of the queue. To retrieve or delete data from the stack we have only one position that is the TOP position.
A queue is an object (an abstract data structure - ADT) that allows the following operations:

- Enqueue: Add an element to the end of the queue
- Dequeue: Remove an element from the front of the queue
- IsEmpty: Check if the queue is empty
- IsFull: Check if the queue is full
- Peek: Get the value of the front of the queue without removing it


empty queue
1
enqueue
2 1
enqueue
2 1
dequeue
FIFO Representation of Queue