Data Structures and Algorithms   
37 | P a g e   
L a b I n s t r u c t o r : R a b i a K h a n   
 Lab 07: Queue in Data Structure   
   
Queue   
A Queue is defined as a linear data structure that is open at both ends and the operations are performed in   
First In First Out (FIFO) order.   
   
We define a queue to be a list in which all additions to the list are made at one end, and all deletions from   
the list are made at the other end. The element which is first pushed into the order, the operation is first   
performed on that.   
   
   
   
Implementation:   
Like stacks, Queues can also be represented in an array: In this representation, the Queue is implemented   
using the array. Variables used in this case are   
 Queue: the name of the array storing queue elements.   
 Front: the index where the first element is stored in the array representing the queue.   
 Rear: the index where the last element is stored in an array representing the queue.   
   
How to implement Queue using Array?   
 To implement a queue using an array,   
 Create and array arr of size n and   
 Take two variables front and rear both of which will be initialized to 0 which means the queue is   
currently empty.   
 Element   
Rear is the index up to which the elements are stored in the array and   
Front is the index of the first element of the array.   
Enqueue:   
 Addition of an element to the queue.   
 Adding an element will be performed after checking whether the queue is full or not.   
 If rear < n which indicates that the array is not full then store the element at arr[rear] and increment   
rear by 1 but if rear == n then it is said to be an Overflow condition as the array is full.   
Dequeue:   
 Removal of an element from the queue.   
 An element can only be deleted when there is at least an element to delete i.e. rear > 0.   
 Now, the element at arr[front] can be deleted but all the remaining elements have to shift to the   
left by one position in order for the dequeue operation to delete the second element from the left   
on another dequeue operation.

Data Structures and Algorithms   
38 | P a g e   
L a b I n s t r u c t o r : R a b i a K h a n

Data Structures and Algorithms   
39 | P a g e   
L a b I n s t r u c t o r : R a b i a K h a n   
 LAB ASSESSMENT   
   
   
   
   
   
   
   
Print Job Management System   
Learning Outcomes:   
   
Understanding of queue data structures and their applications.   
   
Implementation of priority queues for job prioritization.   
This project provides a practical application of data structures and algorithms in a real-world   
scenario, giving students hands-on experience with queue-related concepts.   
Tasks:   
The goal of this project is to create a print job management system that efficiently handles print jobs   
submitted by users. The system will prioritize print jobs based on their size, and users will be able to   
check the status of their print jobs.   
1. Basic Queue Implementation:   
o   
Implement a basic queue to manage the print jobs.   
o   
Each print job should have a unique identifier and details about the number of pages to be   
printed.   
2. Priority Queue:   
o   
Implement a priority queue to prioritize print jobs based on the number of pages.   
o   
Smaller print jobs should have higher priority.   
3. Job Submission:   
o   
Create a function to allow users to submit print jobs to the system.   
o   
Each submitted job should be added to the queue.   
4. Print Job Processing:   
o   
Implement a function to process print jobs from the queue.   
o   
The system should dequeue jobs based on priority and simulate the printing process.   
5. Job Status Checking:   
o   
Allow users to check the status of their print jobs.   
o   
Display information such as position in the queue and estimated print time.   
6. Circular Queue Implementation:   
o   
Implement a circular queue to handle scenarios where the print jobs continue in a loop.   
Bonus Tasks   
9. Error Handling:   
o   
Implement error handling for cases like attempting to check the status of a non-  
existent print job or submitting a print job when the queue is full.   
10. Logging and Statistics:   
o   
Implement a logging mechanism to record the history of print jobs processed.   
o   
Display statistics such as average print time, total number of print jobs, etc.   
Note  
   
After going through questions and answers related to the tasks, submit a separate Word document which should   
include all the results of the tasks. Task should be attempted and submitted within the lab.   
First page should contain your Full name, Registration #, Course name and Date