Course code: CSL 201 Course Name : Data Structures Lab

Faculty In Charge: Dr Binu V P

L-T-P-Credits 0-0-3-2

Pre-requisite: CST 201 Data Structures, EST 102 C programming skills.

Operating System to Use in Lab : Linux

Compiler/Software to Use in Lab : gcc

Programming Language to Use in Lab : Ansi C

Preamble: The aim of the Course is to give hands-on experience for Learners on creating and using different Data Structures. Data Structures are used to process data and arrange data in different formats for many applications. The most commonly performed operations on data structures are traversing, searching, inserting, deleting and few special operations like merging and sorting.

Lab Cycle-5

Learning Outcome: Implement various type of Queues.

Date of submission: on or before 28-10-2022

(Write these programs in fair record-show the output in lab and get it signed by the staff in charge. There will be viva voce in every lab.)

1. Queue - Implement using array and Linked List

Queue a First-In-First-Out (FIFO) data structure. In a FIFO data structure, the first element added to the queue will be the first one to be removed. This is equivalent to the requirement that whenever an element is added, all elements that were added before have to be removed before the new element can be invoked. A queue is an example of a linear data structure.

Queues provide services in computer science and operations research where various entities such as data, objects, or events are stored and held to be processed later. In these contexts, the queue performs the function of a buffer.

2.Circlular Queue (Ring Buffer)- Implement Using Array and Linked List

A circular buffer or ring buffer is a data structure that uses a single, fixed-size buffer as if it were connected end-to-end. This structure lends itself easily to buffering data streams.

A key advantage of a circular buffer is that it has a static size and elements need not be shuffled around when a portion of the buffer is used. This means that only new data is written to the buffer and the computational cost is independent of the length of the buffer.

It has got number of applications in Computers and communication like implementing a transmission flow control protocol.

3.Priority Queue -Implement Using Array or Linked List

One can imagine a priority queue as a modified queue, but when one would get the next element off the queue, the highest-priority one is retrieved first.

There are a variety of simple, usually-inefficient ways to implement a priority queue. They provide an analogy to help one understand what a priority queue is:

Sorted list implementation: Like a checkout line at the supermarket, but where important people get to "cut" in front of less important people. (O(n) insertion time, O(1) get-next time)

Unsorted list implementation: Keep a list of elements as the queue. To add an element, append it to the end. To get the next element, search through all elements for the one with the highest priority. (O(1) insertion time, O(n) get-next due to search)

In practice, priority queues blend these two approaches, so any operation takes roughly O(log(n)) time or less. To get better performance, priority queues typically use a heap

It has got applications in CPU scheduling and memory management schemes. In a multi user environment the highest priority process must be allocated to the processor.

4.Double Ended Queue –Implement using linked list

A deque (short for double-ended queue—usually pronounced deck) is an abstract list type data structure, also called a head-tail linked list, for which elements can be added to or removed from the front (head) or back (tail).

This differs from the queue abstract data type or First-In-First-Out List (FIFO), where elements can only be added to one end and removed from the other. This general data class has some possible sub-types:

An input-restricted deque is one where deletion can be made from both ends, but input can only be made at one end.

An output-restricted deque is one where input can be made at both ends, but output can be made from one end only.

Both the basic and most common list types in computing, the queues and stacks can be considered specializations of deques, and can be implemented using deques.