**DATA STRUCTURES AND ALGORITHMS : WEEK 7**

**Linked Lists** - certain class of lists may be represented using arrays Lists that are fixed in length.

Representing such kind of linear lists is the **linked list**.

Two types of linked lists:

o **Singly linked list**

o **Doubly linked list**

**Data field** - or simply data is used to contain the value of the element.

**Pointer field**, link, or simply, reference, contains the address of the next node in the list of a computer memory.

All references are the same size no matter what they refer in a given computer/operating system The "Next Node" in the list is called the **SUCCESSOR.**

**Head** – first node in the list contains the “Martin” There is no limit to the size of a singly-linked list Adding a node to a linked list.

**Insert** - usually inserted at the beginning of the lis

**Search** - moves along the list searching for the specified value then prompts you the address of the node

**Delete** – search for the value and delete the data then connects the arrow from the previous link straight across to the following link

**DATA STRUCTURES AND ALGORITHMS : WEEK 8**

**DATA STRUCTURES AND ALGORITHMS : WEEK 9**

Data field and the right pointer field function in the same manner as in singly linked lists left pointer field is used to contain the address of the preceding node in the list or what is known as the **PREDECESSOR**

**Traversal Operation** displayBackward() algorithm start at end until starts of list display data move to previous link.

**Insertion Methods** insertLast() method algorithm make a new link if empty list first newLink old last newLink old last newLink last

**insertAfter() method algorithm** start at beginning until match is found, move to next link not found, return false make a new link if last link newLink null newLink last else not last link newLink old next newLink old next old current newLink old current newLink when found – insertion done

**deleteKey() method algorithm** first item? first old next not first old previous old next last item? old previous last not last old previous old next

**Inserting a Node into a Doubly-Linked List** General procedure Create a new node for the element. Set the data field of the new node to the Value to be inserted. Determine the position of the node in The list based on its value. Insert the node

**DATA STRUCTURES AND ALGORITHMS : WEEK 10**

**Queue** - an ordered list

**Top/rear** – end of the list where all insertion takes place

**Bottom/front** – other end of the list where all deletions take place

This type of processing behavior is called **FIFO (First in First Out)**

With stack, only one element is "visible" at a given time, while on queue, the "visible” element will always be the one at the bottom of the queue

Queues are represented as one-dimensional arrays when:

The maximum size of the queue is known.

The amount of time wasted in shifting the elements after deletion is negligible.

**Queue as Singly-linked list** when the maximum size of the queue is unknown.

**Enqueue** - always at the top of the list

**Operations on Queue**

**Dequeue***-*delete at the bottom of the list

**Peek or Retrieve***-*print or display the element at the bottom of the list

being viewed or retrieve is what you are about to delete or remove and not what you have inserted

**Empty and Full***-* return you an error because the queue isempty

a message that the queue is full when trying to insert another element

**Circular Queue** -Sometimes called as ring buffer elements of the array are “arranged” in a circular manner