EXPERIMENT NO-06

* Write a program to implement a circular queue using arrays.

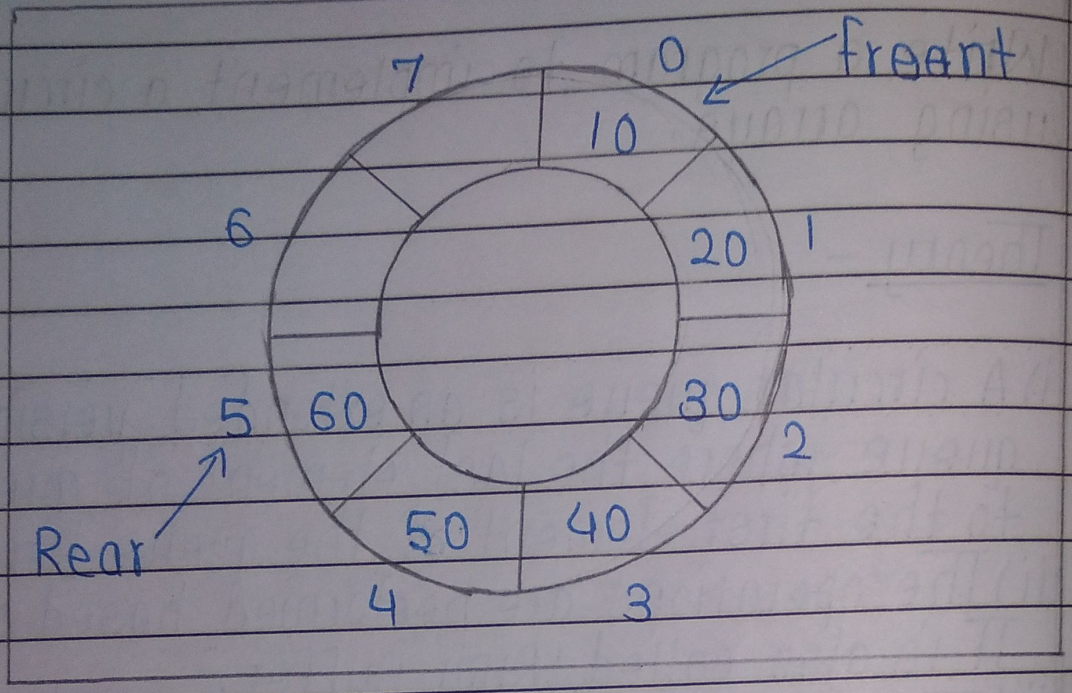
* Theory -

i) A circular queue is an extended version of a normal queue where the last element of queue is connected to the first element of the queue forming a circle.
ii) The operations are performed based on FIFO principle. It is also called 'Ring Buffer'.
iii) In a normal queue we can insert element until queue becomes full.
iv) But once queue become full we can't insert the next element even if there is a space in front of queue.

Operations on circular queue -

i) Front
ii) rear
iii) enQueue ()
iv) deQueue ()
v) display ()

Diagram -



1] Enqueue

i) Increment rear by 1. If rear is equal to n, set rear to 0.
ii) If front is -1, set front to 0.
iii) Set queue [rear] to x.

2] Dequeue -

i) Check if front is -1, if it is return underflow condition. Set x to queue [front]
ii) If front is equal to rear, set front and rear to -1
iii) Otherwise increment front by 1 and if front is equal to n, set front to 0.
iv) Return x.

Conclusion –

i) Circular queue using arrays offers efficient memory usage and fixed capacity.
ii) They allow for efficient insertion and deletion of elements by wrapping around the array.
iii) However, the fixed capacity may limit the number of element that can be stored.

EXPERIMENT NO - 07

* Write a program to implement a double ended queue (dequeue) using arrays.

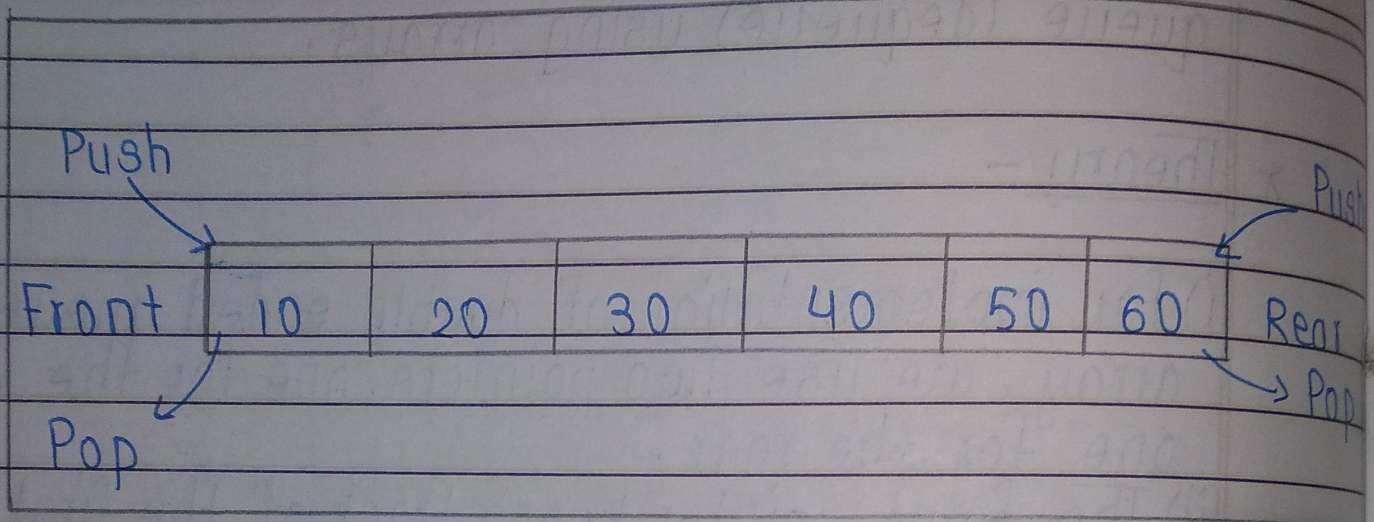
* Theory -

i) For implementation of double ended queue using array, we use two pointers, one for the front and one for the rear.
ii) The front pointer keeps track of the first element and the rear pointer keeps track of the last element.
iii) In double ended queue we can insert and delete the element from both the end i.e from front as well as from rear.
iv) This implementation allows us to efficiently add or remove elements from both ends of the queue using array in C.
v) It provides flexibility and versatility in managing the data structure.

Operation on double ended queue -

i) insert front () - Add the items at front
ii) insert rear () - Add item at rear
iii) delete front () - Delete item from front
iv) delete rear () - Delete item from rear

Diagram -



Conclusion -

i) Overall, the double ended queue (deque) is a data structure that allows element to be added or remov from both the front and back of the queue.

ii) This make it more flexible than a regular queue which can only inserted from the back and delete from front.

iii) Also, array-based queues are faster than last-based queues.

iv) Array-based queues require less memory than list based queues.

EXPERIMENT NO 08

* Write a program to implement the data structure double linked list.

* Theory -

i) Doubly linked list is a complex type of linked list in which a node contains a pointer to the previous as well as the next node in the sequence.
ii) Therefore, in doubly linked list, a node consist of three parts: node data, pointer to the next node in sequence and pointer to the previous node.

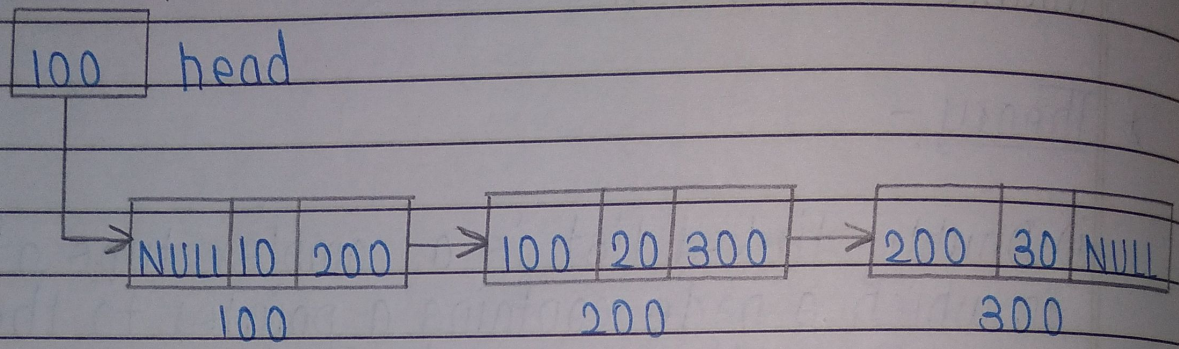
Memory representation of doubly linked list -

i) Doubly linked list consume more space for every node and therefore causes more expensive basic operation such as insertion and deletion.
ii) However, we can easily manipulate the element of a list since the list maintains pointer in both the directions.

Operation on doubly linked list -

i) Insertion ()
ii) Deletion ()
iii) Traversal ()

Diagram -



1] Insertion -

We can insert the node at the beginning, end or at a given position in the doubly linked list. To insert the node, you update the pointers of the previous and next nodes accordingly.

2] Deletion -

We can delete a node from the doubly linked list. Similar to insertion, we update the pointer of the previous and next nodes to remove the node from the list.

3] Traversal -

We can traverse the doubly linked list in either direction starting from the head or tail, and visit each node to perform specific operations or access its data.

Conclusion -

i) Doubly linked list provides flexibility in traversing the list both forward and backward.
ii) Insertion and deletion operations can be easily performed by updating the pointers of adjacent nodes.
iii) Searching for a specific value in a doubly linked list is efficient due to the ability to traverse on both the direction.

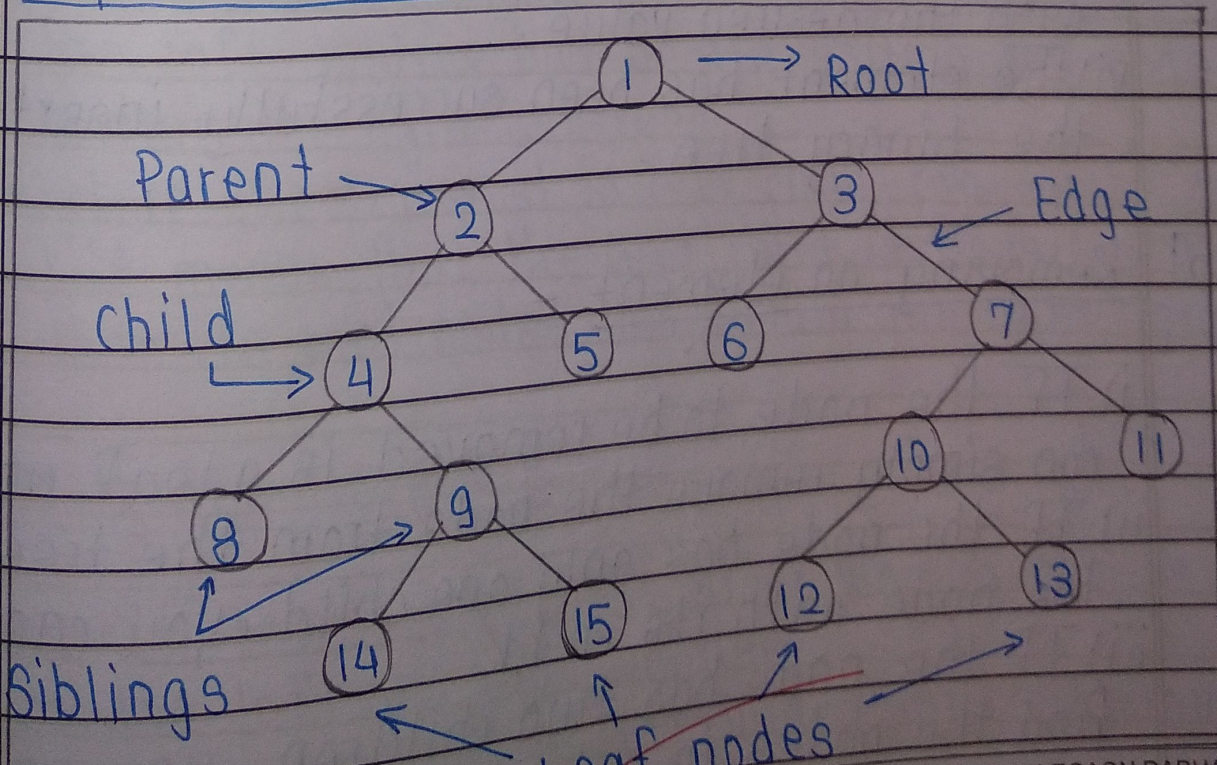
EXPERIMENT NO - 09

* Write a program for creating a binary tree and traversal of binary tree -

* Theory -

i) A binary tree is a tree data structure in which each node can have atmost two children, which are reffered to as the left child or the right child.
ii) The topmost node in a binary tree is called as a root, and the bottom most are called as a leaves.
iii) They can also be used to implement various algorithms such as searching sorting and the graph algorithms.

Representation of binary tree -



Each node in the tree contains the following -

i) Data
ii) Pointer to the left child
iii) Pointer to the right child.

Basic operations on binary tree -

1] Inserting an element -

i) Begin at the root of the binary tree.
ii) Compare the value of the element with the value of the current node.
iii) If the value is less than the current nodes value move to the left otherwise right.
iv) Repeate step 2 and 3 until you find a position.
v) Once an empty spot is found create a newnode with the desired value.
vi) The element has been successfully inserted into the binary tree.

2] Removing an element -

i) If the node to be removed is a leaf node, you can simply remove the node from the tree.
ii) If the node has only one child, you can replace the node with its child.
iii) If the node has two children, you need to find the nodes in-order successor.

iv) Replace the nodes value and then recursively delete the in-order successor from the right subtree.

Types of binary tree traversal -

1] Preorder traversal -

i) Visit the root node before visiting any nodes inside the left or right subtree.
ii) Here, the traversal is root - left child - right child.
iii) It means that the root node is traversed first then its left child and finally the right child.

2] Inorder traversal -

i) Visit the root node after visiting all nodes inside the left subtree but before visiting any node within the right subtree.
ii) Here, the traversal is left child - root - right child.
iii) It means that the left child is traversed first then its root nodes and finally the right child.

3] Postorder traversal -

i) Visit the current node after visiting all the nodes of the left and right subtree.
ii) Here the traversal is left child - right child - root.
iii) It means that the left child has traversed first

then the right child and finally its root node.

conclusion -

In conclusion, binary trees provide a hierarchical way of organizing and accessing data efficiently. They have various application in algorithm and data manipulation task.

EXPERIMENT NO-10

* Write a program for selection sort -

* Theory -

i) Selection sort is a simple and efficient sorting algorithm that works by repeatedly selecting the smallest element from the unsorted portion of the list and moving it to the sorted portion of the list.
ii) The algorithm repeatedly select the smallest (or largest) element from the unsorted portion of the list and swaps it with the first element of the unsorted part.
iii) This process is repeated for the remaining unsorted portion until the entire list is sorted.

Steps for selection sort -

1] Initialization -

The algorithm divides the array into 2 parts - sorted and unsorted. Initially, the sorted part is empty and the unsorted array is the enitire array.

2] Finding the minimum -

It searches the unsorted part to find the minimum element and swap it with the first element in the

unsorted part, thus expanding the sorted part.

3] Updating pointers -

The boundaries of the sorted and unsorted part adjusted. The sorted part grows and the unsorted part shrink.

4] Repeat -

Step 2, 3 repeated until the entire array is sorted.

Advantages of selection sort algorithm -

i) Simple and easy to understand
ii) Work well with small dataset

Disadvantage of selection sort algorithm -

i) Selection sort has a time complexity of $O(n^2)$ in the worst and average case.
ii) Does not work well on large dataset.

Conclusion -

i) Despite its inefficiently, selection sort can be useful for sorting small list.
ii) It is also a good algorithm to use if you need to sort a list in place, without using any additional memory.