Summary

Sessions (16-02-2023)

- **Queue : Using Doubly Linked List**

- Create Node class for node and Queue class for managing their methods.

```
class Node {
      public:
            int data;
            Node *prev;
            Node *next;
};

class Queue {
      private:
            Node *head, *last;
      public:
            Queue();
            void enqueue(int newData);
            void Display();
            bool isEmpty();
            int dequeue();
};
```

- Create a Head Node.

```
Queue::Queue() {
        head = new Node;

        head -> data = NULL;
        head -> prev = NULL;
        head -> next = NULL;

        last = head;
}
```

- **Enqueue** : This operation adds a new node after the last node and moves the last node to the next node.

- Create Enqueue method :

```
void Queue::enqueue(int newData) {
        Node *newNode = new Node;
        newNode -> data = newData;
        newNode -> prev = last;
        newNode -> next = NULL;
        last -> next = newNode;
        last = newNode;
}
```

- Time complexity & Space complexity is O(1) ie. constant

- **Dequeue** : This operation removes the front node and moves the front to the next node.

- Create Dequeue method :

```
int Queue::dequeue() {
      Node *temp = head -> next;
      int adata = temp -> data;
      if (temp -> next != NULL) {
            head -> next = temp -> next;
            temp -> next -> prev = head;
      }
      else {
        head -> next = NULL;
      }
      delete temp;
      return adata;
}
```

- Time complexity & Space complexity is O(1) ie. constant

- Create Display method :

```
void Queue::Display() {
      Node *temp = head;
      while(temp -> next != NULL) {
            cout << temp -> next -> data << endl;
            temp = temp -> next;
      }
}
```

- Time complexity is O(n) and Space complexity is O(1).

- Create isEmpty method :

```
bool Queue::isEmpty() {
      if(head -> next == NULL) {
            return true;
      }
      else
            return false;
}
```

- Time complexity & Space complexity is O(1) ie. constant