LINKED LIST

Define a class **Node** with following specification: value

nextNode

**Data members:**

value : IT store a numeric value

nextNode : It store the reference of an object of Node class

**Constructor:**

**Node()** : It initialize data members  **value** and **nextNode** with following values **0** and **null** respectively

**Function**

setNextNode(Node n) : This function will initialize **nextNode** with Node reference **n**

setValue(int val) **:** this function assign **val** to **value** data member of the object

Define a class LinkedList to implement a singly LinkedList

**Data members:**

HEAD : Hold the reference to the starting node (HEAD) of the linked list

TAIL : Hold reference of Tail of the linked list

**Function:**

void addNode ( int num) : to append a new node to the tail of existing linked list

Q1 Write an Algorithm OR a Method to add a node at the end of an existing linked list. The method declaration is as follows:  
void add node (Node start, int num)

Q2 Write an Algorithm OR a Method to count the number of nodes in the linked list. The method declaration is given below:  
int count (Node ptr-start)

Q3 A linked list is formed from the objects of the class: [4]

class Nodes

{

int num;

Node next;

}

Write an Algorithm OR a Method to print the sum of nodes that contains only odd integers of an existing linked list.  
The method declaration is as follows:  
void NodesCount (Nodes starPtr)

QUEUE

A doubly queue is a linear data structure which enables the user to add and remove integers from either ends, i.e. from front or rear. Define a class Dequeue with the following details: [10]  
Class name: Dequeue  
Data members/instance variables:  
arr[ ]: array to hold up to 100 integer elements  
lim: stores the limit of the dequeue  
front: to point to the index of the front end  
rear: to point to the index of the rear end  
Member functions:  
Dequeue(int 1): constructor to initialize the data members lim = 1; front = rear = 0  
void addfront(int val): to add integer from the front if possible else display the message (“Overflow from front”) voidaddrear(intval): to add integer from the rear if possible else display the message (“Overflow from rear”)  
int popfront(): returns element from front, if possible otherwise returns – 9999  
int poprear(): returns element from rear, if possible otherwise returns – 9999  
Specify the class Dequeue giving details of the constructor (int), void addfront(int), void addrear (int, popfront ( ) and int poprear ( ). The main function and algorithm need not be written.