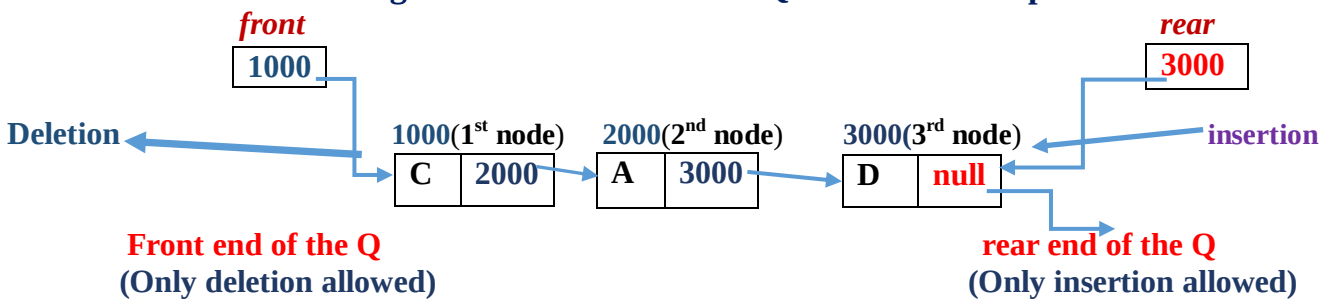
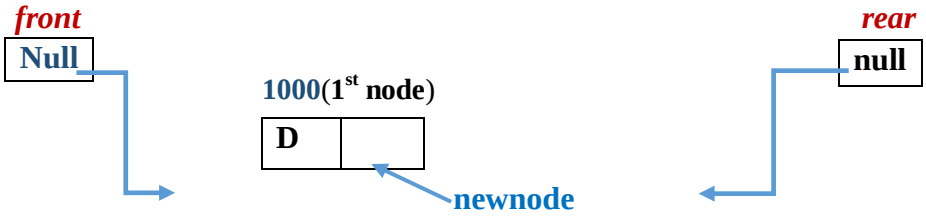
```
class QNODE
{
        char info;
        QNODE link;                    /*Structure of a node in single linked list*/
}
/* The above  class 'NODE' will be used to create nodes , nodes are nothing but
class type objects that you were creating in ICP by using new operator and
constructor call */
/*By using the above class declaration to create nodes of a linear queue, each node
will hold a single character value in the information part.*/
public class Q_LL_DEMO
{       static QNODE front=null; /* creates an empty Q initially */
        static QNODE rear=null;  /* creates an empty Q initially */
       public static void main(String[] args)
       {
            Scanner sc=new Scanner(System.in);
            char ch;
            int opt;
            char data;
        /*The following do…while loop implements the concept of menu driven program*/
            do
            {
           /* Display the menu consisting of different operations on single linked list*/
                    System.out.println("1. Insert_Q 2. Delete_Q ");
                    System.out.println("3. Traverse_Q");

                    System.out.println("Enter your option");
                     opt=sc.nextInt();
                   switch(opt)
                   {
                         case 1:
                            System.out.println("Enter the new item to inserted onto Q:");
                                data=sc.next().charAt(0);
                                insert_Q(data);
                                break;
                          case 6:
                                delete_Q();
                                 break;
                          case 7:
                                traverse_Q();
                                 break;
                            default:
                                  System.out.println("Invalid option" );
                      }   /* End of switch */
```

```
            System.out.println("\nDo you want to perform another operation(y/n)");
            ch=sc.next().charAt(0);
        }while(ch=='y'|| ch== 'Y');  /* End of do---while loop */
    }  /* End of MAIN method */
```

/*Fig: 1 Structure of a linear Q in linked list implementation*/

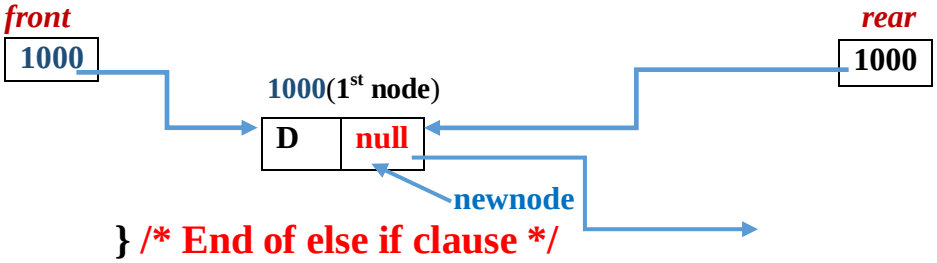



```
/*INSERT_Q METHOD: Inserts a new node at front end of the Q*/
public static void insert_Q(char new_item) /*new_item is value of the new node*/
{
        QNODE newnode=new QNODE();
        newnode.info=new_item;
        if ( newnode == null )
        {    System.out.println("Memory is full, u can't create new nodes");
             Return ;
         }
         else  if ( rear == null)   /* checks if the Q is empty initially*/
         {    /*Fig: 2 After creating the 1st node when the Q is initially empty*/
```



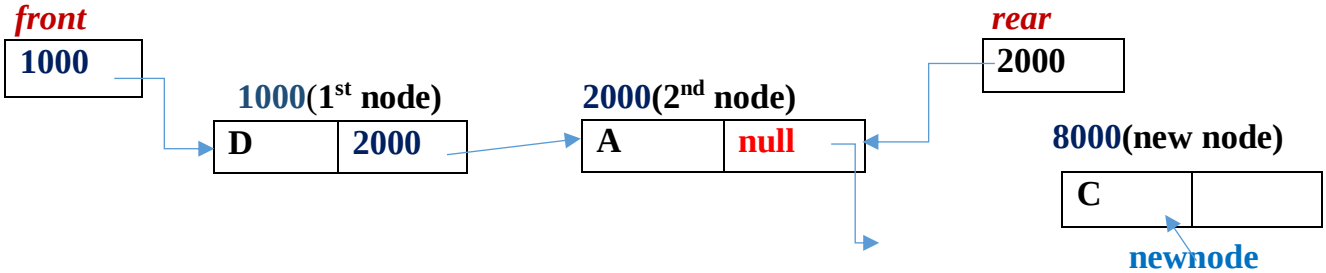

```
                front=rear=newnode;
                newnode.link=null;
```

/*Fig: 3 After attaching the new node at rear end when the Q is initially empty*/

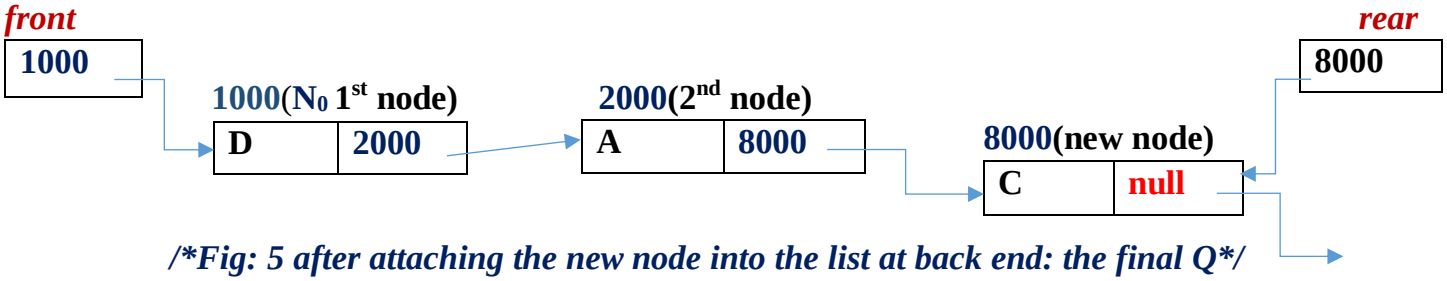



```
         } /* End of else if clause */
        else
        {
```

/*Fig: 4 before attaching the new node into the list at rear end when the Q is not empty*/

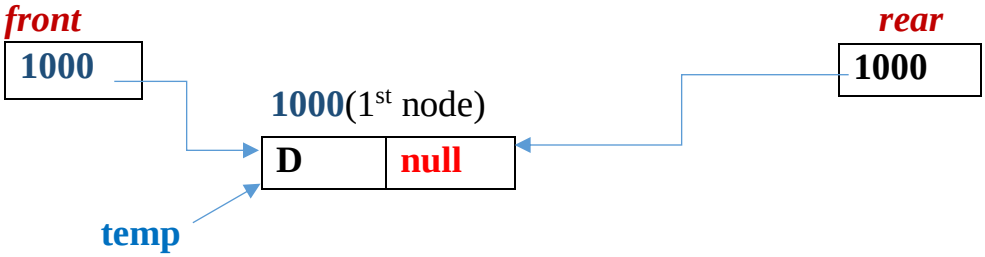


```
rear.link=newnode;
rear=newnode;
newnode.link=null;
```



*/*Fig: 5 after attaching the new node into the list at back end: the final Q*/*

```
        } /*End of else clause*/
} /* End of insert at beginning of the list method */

/* DELETION FROM QUEUE*/
public static void delete_Q()
{
        if(start == null)
        {
            System.out.println("list is empty ....");
            return;
         }

          QNODE temp=front; /*←Stores 1st nodes address in temp if Q is not empty*/

          if ( front == rear) /*← checks whether the Q contains only one node?*/
          {
```

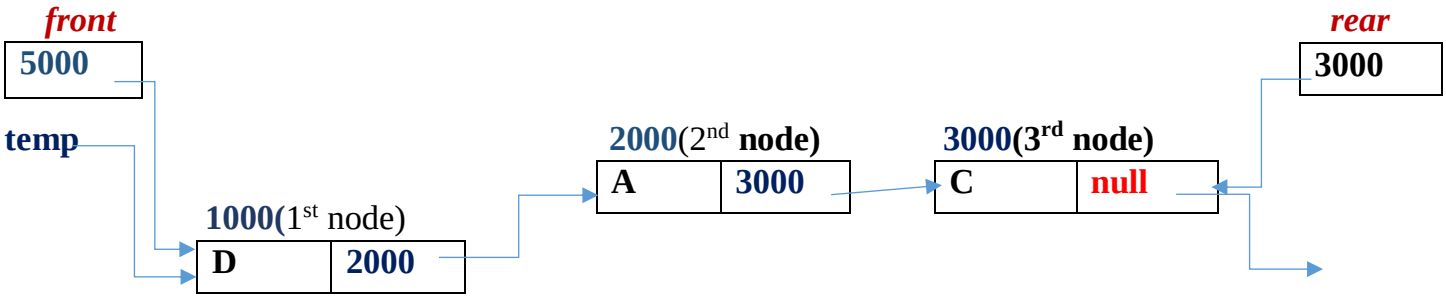


(Figure: 6 Before removal of the node when the Q contains only one node)

```
            front= rear=null; /*← set front and rear to null to make the Q empty */
```

/ (Figure: 7 After removal of the only node : the final empty Q */*

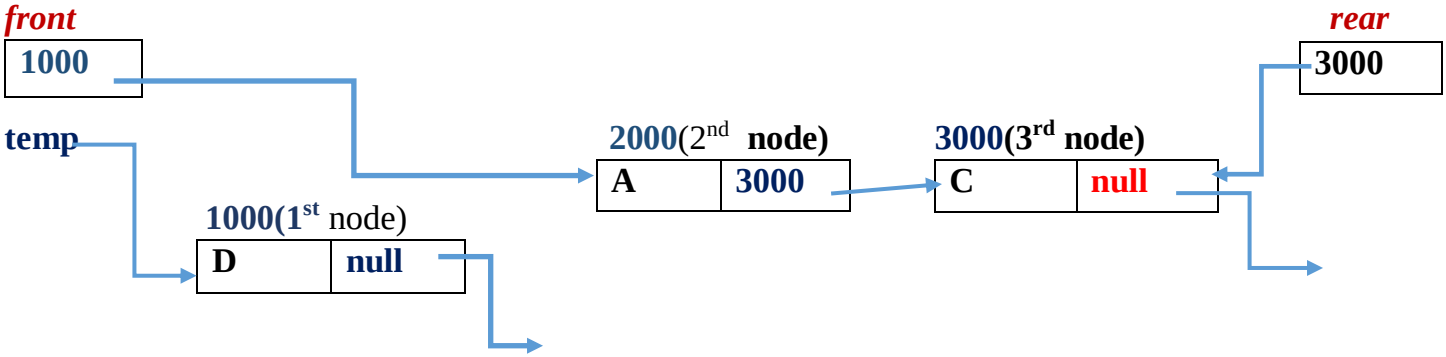
```
          } /* End of if clause*/
           else
           {    /* this else clause is executed if the Q contains more than one node*/
```



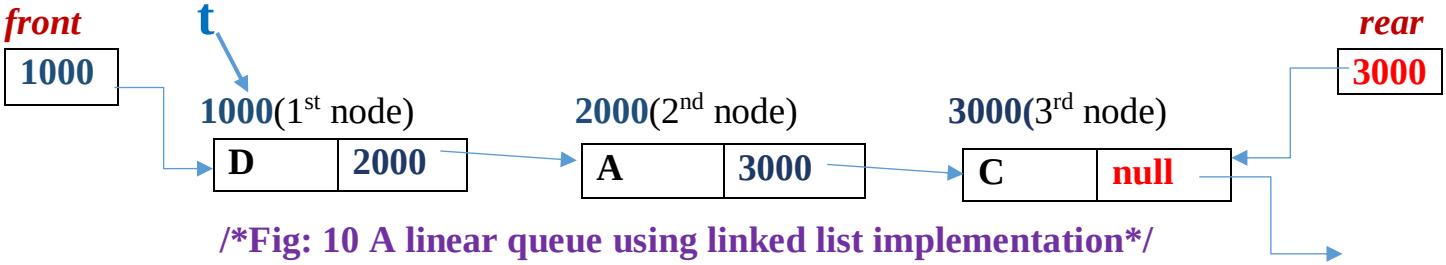
*/*Figure: 8 Before removal of the first node: temp refers to 1st node to be deleted*/*

```
front=front.link;
temp.link=null;
```

/*Fig: 9 After the execution of above statements the list will be as follows*/




```
    } /* End of else clause*/
     System.out.println("The deleted node is:" + temp.info); /* prints  D*/
}/* End of Delete method */

/* TRAVERSE QUEUE METHOD*/
public static void traverse_Q()
{
      If(front==null)  /*← Checks whether the list is empty or not */
      {
          System.out.println("queue is empty");
          return;
       }
      else
      {
           System.out.println("\n The queue you have created is…..\n");
```



/*Fig: 10 A linear queue using linked list implementation*/

```
     QNODE t=front; /*← t refers to 1st node before the while loop starts execution*/
      while(t != null) /*← checks whether we have reached end of the list?*/
      {
           System.out.print(t.info + " → "); /*←prints info part of current node*/
```

```
                t = t.link; /*←moves the reference variable t to the next node */
            }/*End of while loop*/

            System.out.println(); /*prints a new line*/

        }/*End of else clause*/
    }/* End of traverse_Q method */

}  /* END OF Q_LL_DEMO CLASS */
```