18ES611 Embedded System Programming

Sarath tv

The Queue Data Structure in C

Queues are data structures that, like the stack, have restrictions on where you can add and remove elements. To understand a queue, think of a cafeteria line: the person at the front is served first, and people are added to the line at the back.

Thus, the first person in line is served first, and the last person is served last. This can be abbreviated to **F**irst **I**n, **F**irst **O**ut (**FIFO**).

The cafeteria line is one type of queue. *Queues are often used* in programming networks, operating systems, and other situations in which *many different processes must share resources* such as CPU time.

One bit of terminology: the addition of an element to a queue is known as an **enqueue**, and removing an element from the queue is known as a **dequeue**.

Although the concept may be simple, **programming a queue is not as simple as programming a stack**.

Let's go back to the example of the cafeteria line. Let's say one person leaves the line. Then what? Everyone in line must step forward, left?

API

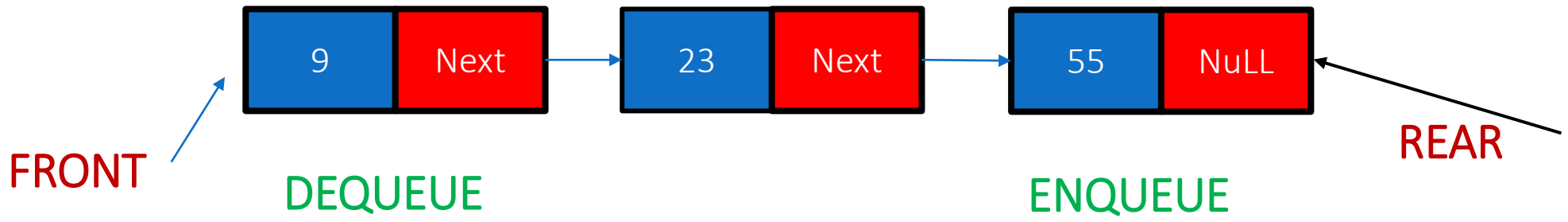
Basic idea:

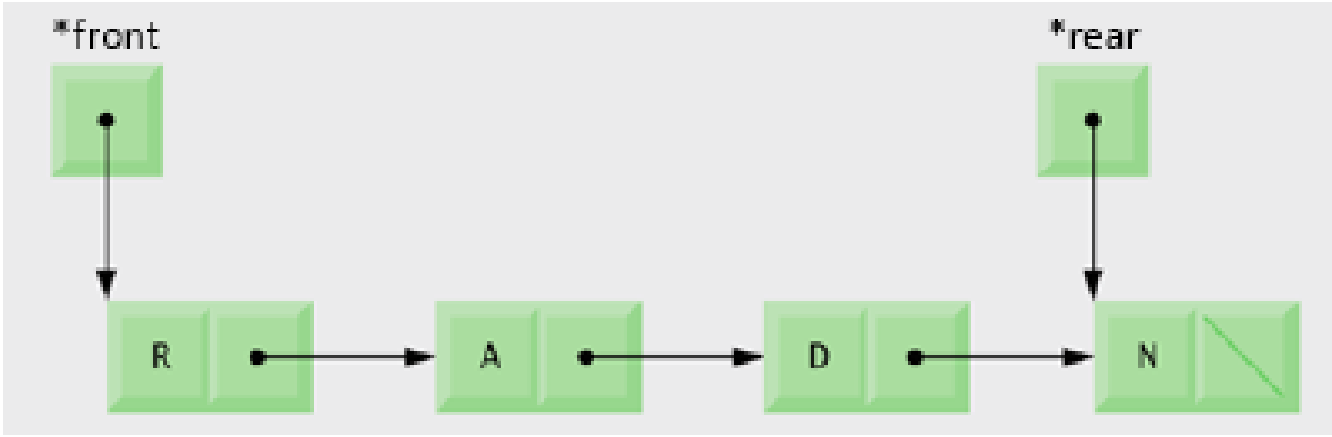
–Create a linked list to which items would be added to one end and deleted from the other end.

–Two pointers will be maintained: Y ???

- One pointing to the beginning of the list (point from where elements will be deleted).
- Another **pointing to the end of the list** (point where new elements will be inserted).

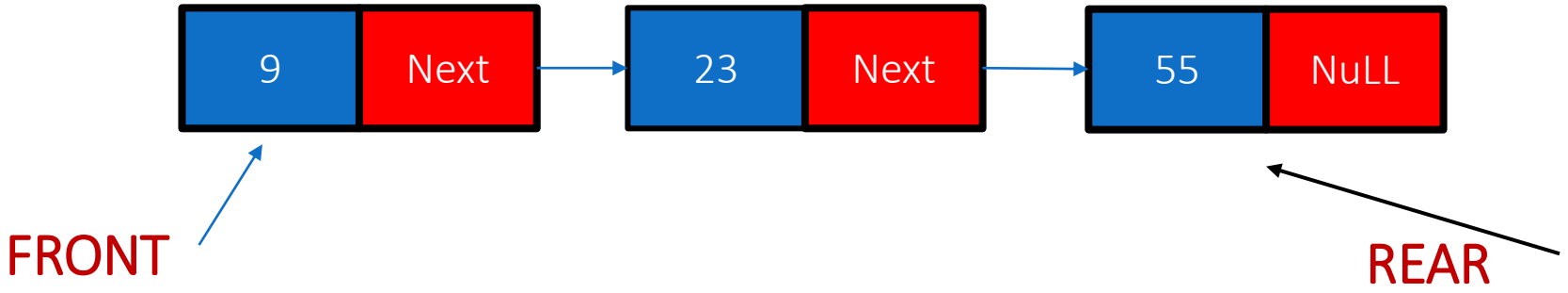
Front = NULL implies ????

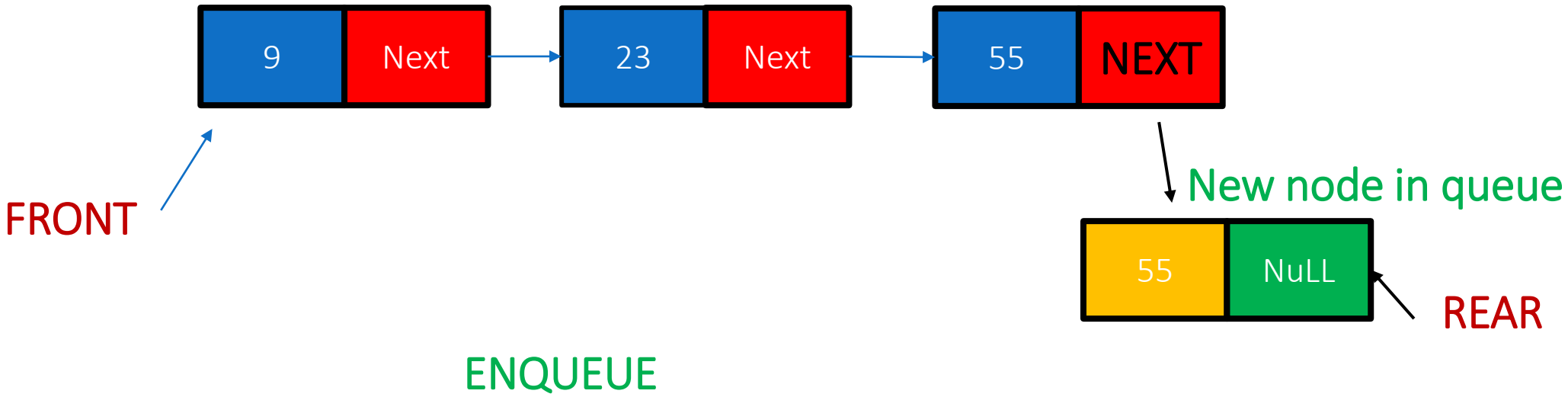





```
typedef struct node
{
int data;
struct node *link;
} NODE

NODE *front, *rear; //*front and *rear are 2 variables
of type NODE
front = rear = NULL;
```

Enqueue





ENQUEUE

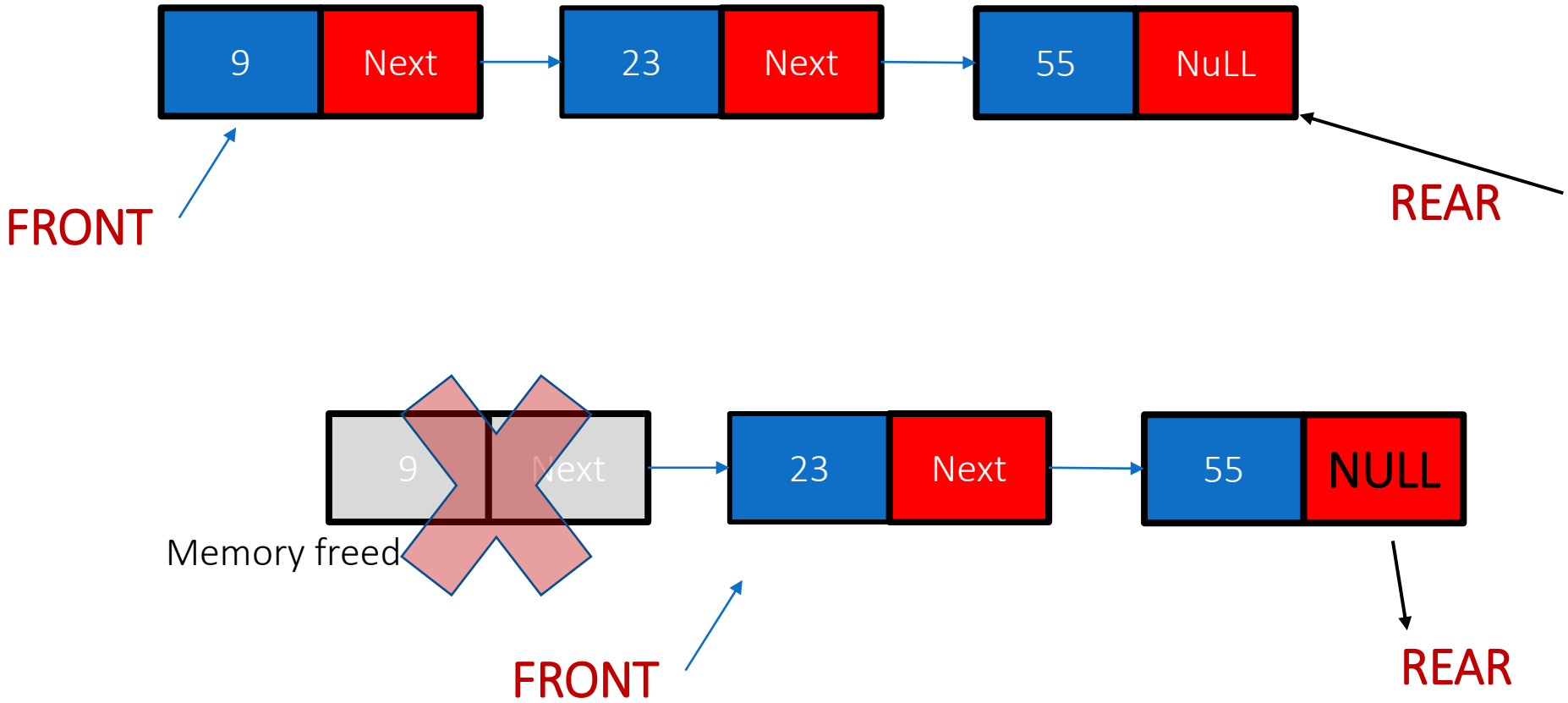
```
void enqueue(int data)
{
    if (rear == NULL)
    {
        rear = (struct node *)malloc(1*sizeof(struct node));
        rear->ptr = NULL;
        rear->info = data;
        front = rear;
    }
    else
    {
        temp=(struct node *)malloc(1*sizeof(struct node));
        rear->ptr = temp;
        temp->info = data;
        temp->ptr = NULL;

        rear = temp;
    }
    count++;
}
```

Empty queue

NON Empty queue

Tracks the total number of nodes in ques

Dequeue



Dequeue

```
void dequeue()
{
    tempfront = front;

    if (tempfront == NULL)
    {
        printf("\n Error: Trying to display
elements from empty queue");
        return;
    }
    else
        if (tempfront ->ptr != NULL)
        {
            tempfront = tempfront ->ptr;
            printf("\n Dequed value : %d", front-
>info);
            free(front);
            front = tempfront;
        }
        else
        {
            printf("\n Dequed value : %d", front-
>info);
            free(front);
            front = NULL;
            rear = NULL;
        }
        count--;
}
```

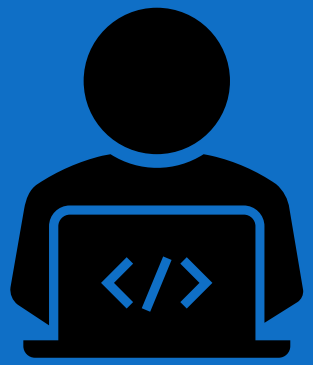
Empty

many node

One node

```
void Display()
{
NODE *t;
if (front == NULL)
printf("Empty Queue\n");
else
{
t = front;
printf("Front->");

while (t)
{
printf("[%d]->", t->data);
t = t->link;
}
printf("Rear\n");
}
```

int frontelement()



THANK YOU!!!!!