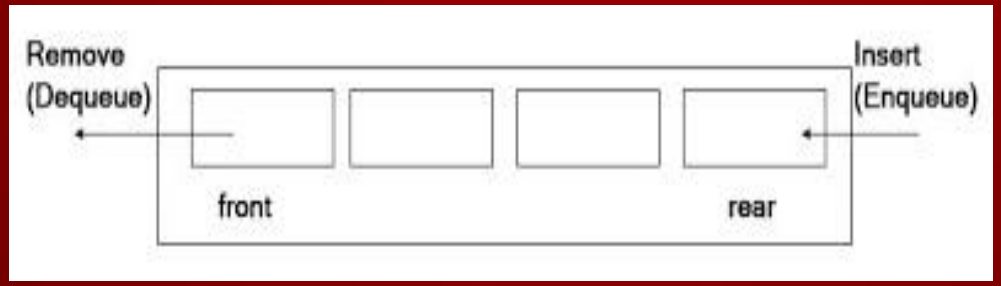
โครงสร้างข้อมูลแบบ QUEUE

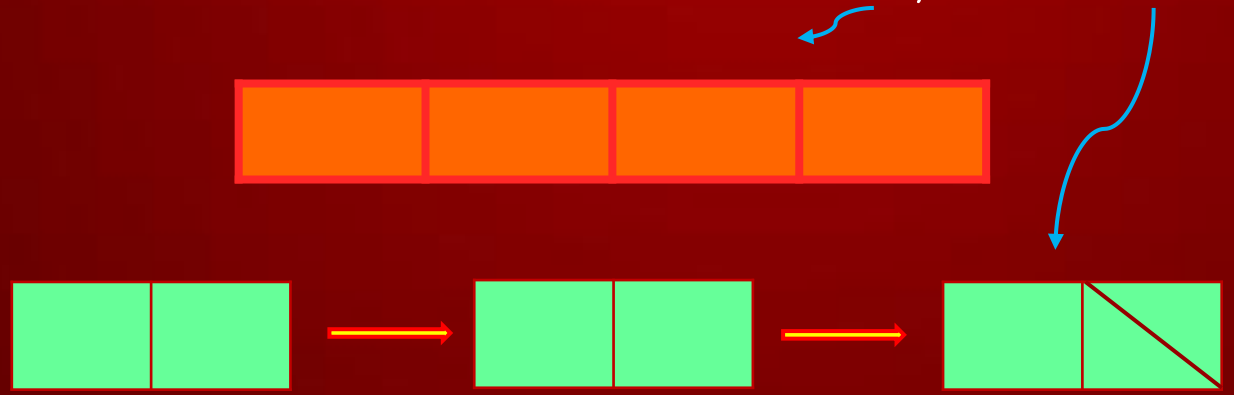
โครงสร้างข้อมูลแบบ QUEUE

- คิว(Queue)เป็นโครงสร้างข้อมูลแบบเชิงเส้นหรือลิเนียร์ลิสตซึ่งการเพิ่มข้อมูลจะกระทำที่ปลายข้างหนึ่งซึ่งเรียกว่าสวนท้ายหรือเรียร์ (rear)และการนำข้อมูลออกจะ กระทำที่ปลายอีกข้างหนึ่งซึ่งเรียกวา ส่วนหน้า หรือฟรอนต์(front)ลักษณะการทำงานของคิวเป็นลักษณะของการเข้าก่อน ออกก่อนหรือที่เรียกว่า FIFO (First In First Out)



QUEUE

- คุณสมบัติที่สำคัญ
  - เป็นโครงสร้างข้อมูลแบบต่อเนื่อง
  - มีการทำงานแบบ First in First out (FIFO)
  - จะต้องทำงานเพิ่มทางด้านข้อมูลทางท้าย(rear)ของคิว และลบข้อมูลทางด้านหน้า (Front) ของคิวเท่านั้น
  - การ implement การทำงานแบบคิว อาจใช้ Array หรือ Linked List

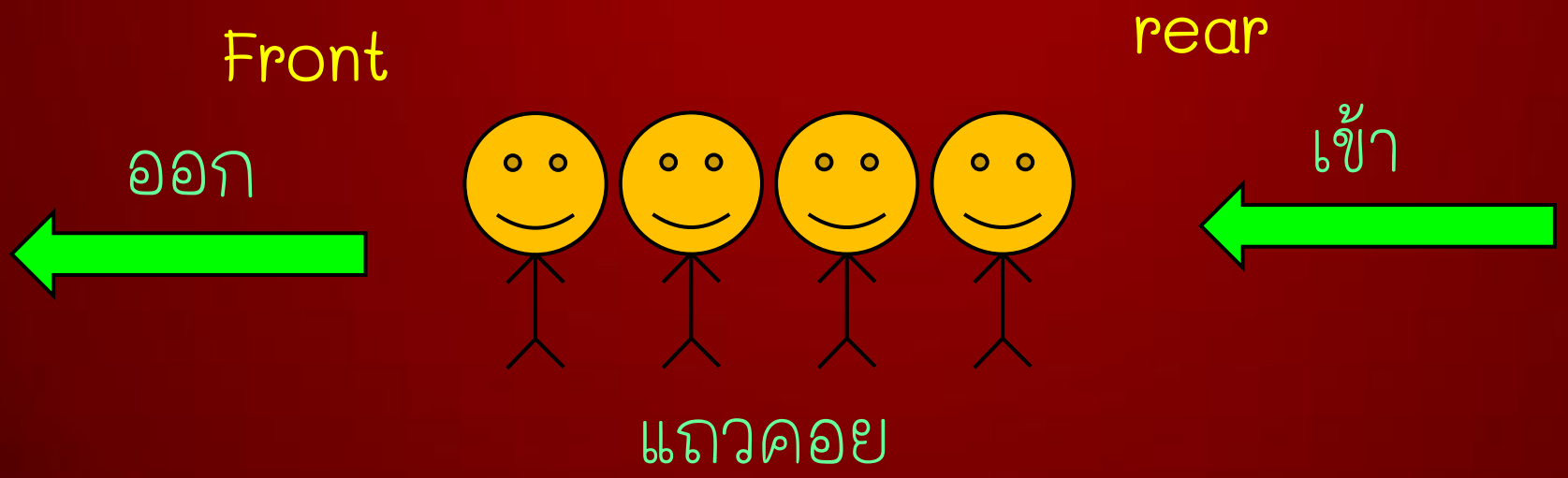


QUEUE

- ประโยชน์ของ Queue
  - ประโยชน์ของคิว ลักษณะของคิวเหมือนแถวคอย หรือการเข้าคิวรอซื้อของ
  - เป็นกลไกสำคัญสำหรับระบบปฏิบัติการ
  - การส่งงานไปยังเครื่องพิมพ์ในระบบคอมพิวเตอร์
  - ระบบการจัดการจราจรทางอากาศ

QUEUE

- การสร้าง Queue
- ใช้ pointer 2 ตัวในการลำดับคิว คือ front และ rear
- front คือตำแหน่งที่ข้อมูลออก จะอยู่ด้านหน้าคิว
- Rear คือตำแหน่งที่ข้อมูลเข้า จะอยู่ด้านท้ายของคิว



QUEUE

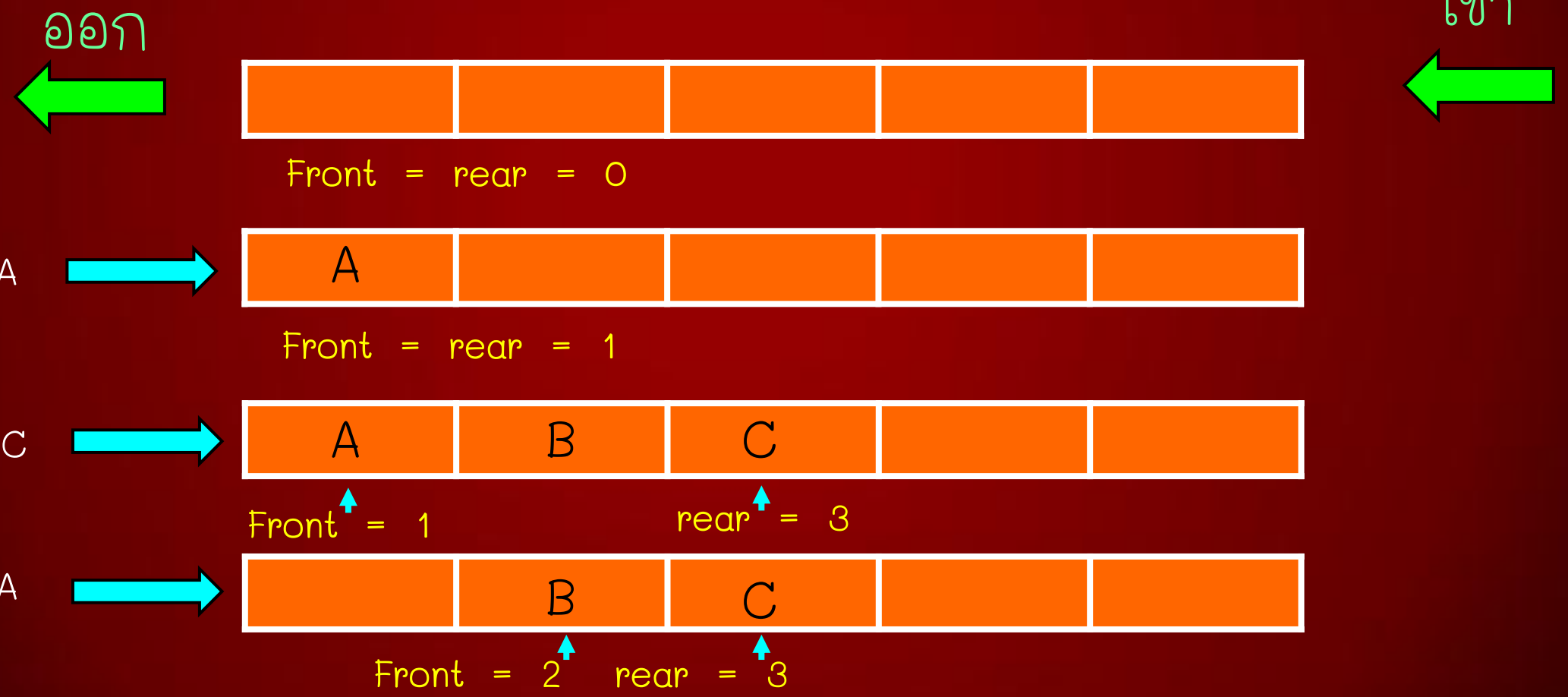
- การปฏิบัติการ Queue
- การเพิ่มข้อมูลเข้าไปในคิว(enqueue) หรือการ insert
- การนำข้อมูลออกจากคิว (dequeue) หรือการ delete

QUEUE IMPLEMENTATION

- Array
- Linked List

Queue : Array Implementation

ตัวอย่างการแทนคิวด้วย Array , size = 5



Queue : Array Implementation

- การเพิ่มข้อมูลเข้าในคิว(enqueue)
- ก่อนนำสมาชิกเข้าคิว ต้องตรวจสอบว่าคิวเต็มหรือไม่ โดยที่ ถ้า rear = maxQ แสดงว่าคิวเต็ม (เมื่อ maxQ คือขนาดของคิว)
- ถ้าคิวเต็ม (full) จะไม่สามารถนำข้อมูลเข้าได้อีก จะทำให้เกิดการล้นของคิว (queue overflow)
- การนำสมาชิกเข้าในคิว จะเข้าทางด้านท้ายคิว (rear)
- ดังนั้นการนำสมาชิกเข้าคิว จึงเป็นการเพิ่มค่าพอยน์เตอร์ rear
- หากมีสมาชิกในคิวเพียงค่าเดียวพอยน์เตอร์ rear และ front จะเท่ากัน

Queue : Array Implementation

```
INSERT-Q Algorithm
  front = rear = 0
  if ( rear == maxQ)
          write "Overflow"
  else
     if(front==0)
          front ⟵ 1
          rear ⟵ 1
          Q[rear] ⟵ Item
  else
          rear ⟵ rear + 1
          Q[rear] ⟵ Item
```

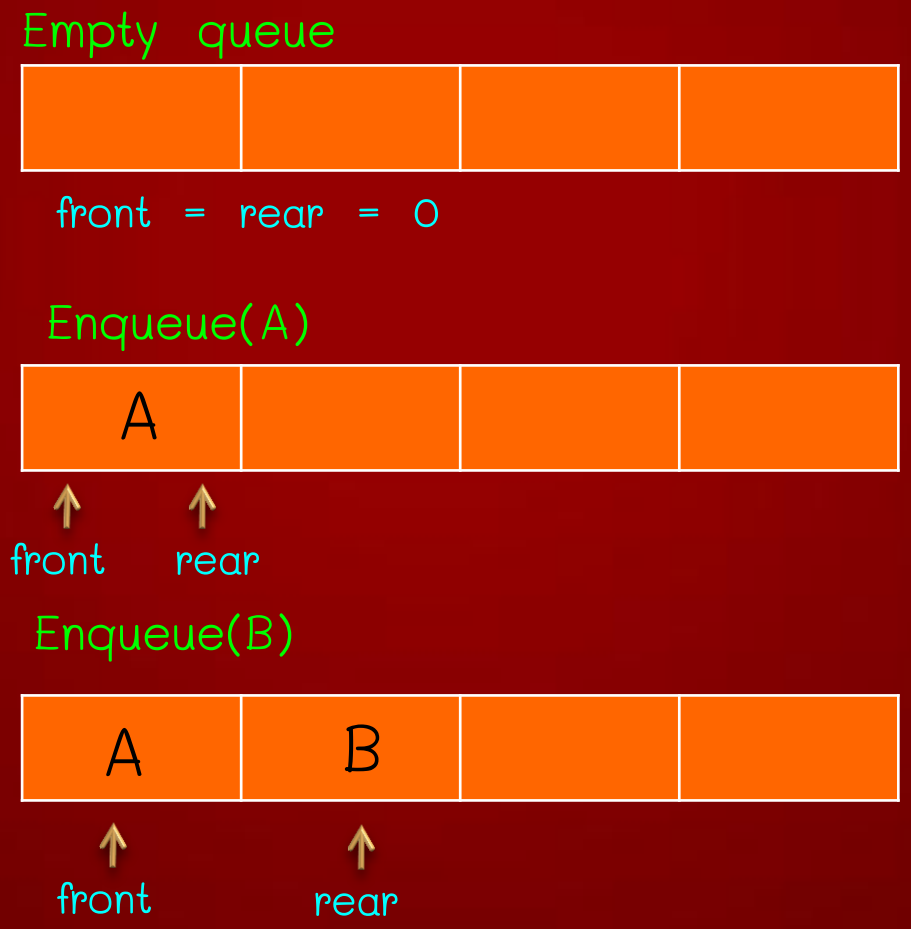
ตรวจสอบว่าคิวเต็มหรือไม่

ตรวจสอบว่าคิวเต็มหรือไม่
ถ้า front = 0 แสดงว่าคิวว่างเปล่า
นั่นคือยังไม่มีข้อมูลในคิว

หมายเหตุ Q คือชื่อคิว maxQ คือขนาดของคิว
Item คือข้อมูลที่ต้องการเพิ่มเข้าไปในคิว

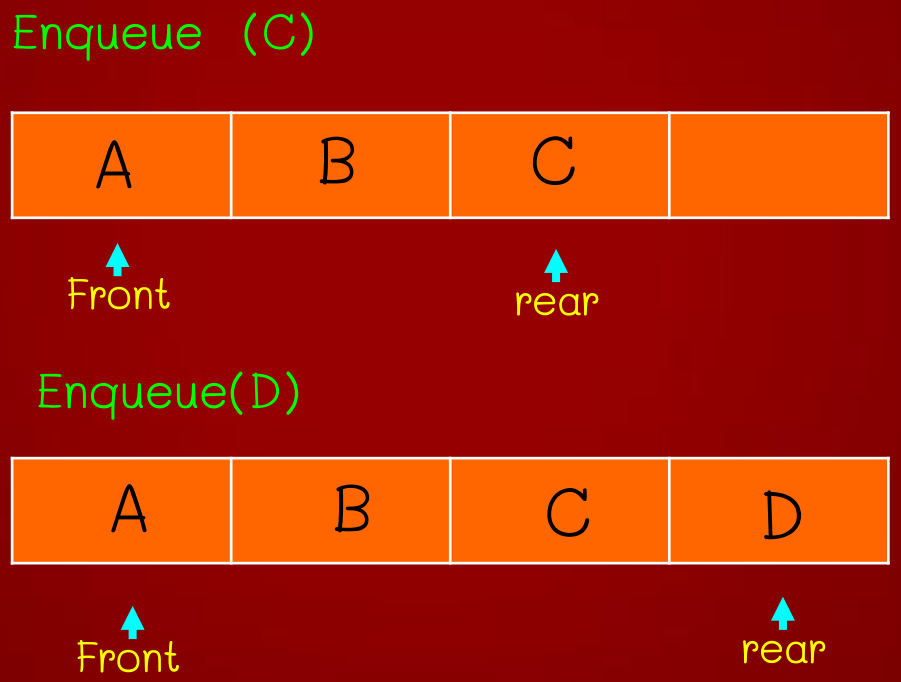
Queue : Array Implementation

- การนำสมาชิกเข้าคิว (enqueue)



Queue : Array Implementation

- การนำสมาชิกเข้าคิว (enqueue) ต่อ



Error : Overflow

Queue : Array Implementation

- การนำข้อมูลออกจากคิว (dequeue)
  - ก่อนนำสมาชิกออกจากคิว ต้องตรวจสอบดูก่อนว่าคิวว่างเปล่าหรือไม่ โดยเงื่อนไขการตรวจสอบคือ front = rear = 0
  - ถ้าคิวว่าง จะไม่สามารถนำสมาชิกออกไม่ได้ จะเกิดปรากฏการณ์ขาดแคลนคิว (queue underflow)
  - การนำสมาชิกออกจากคิว จะทำด้านหน้าคิว (front)
  - ดังนั้นการนำสมาชิกออกจากคิวจึง เป็นการเพิ่มค่าพอยน์เตอร์ front

Queue : Array Implementation

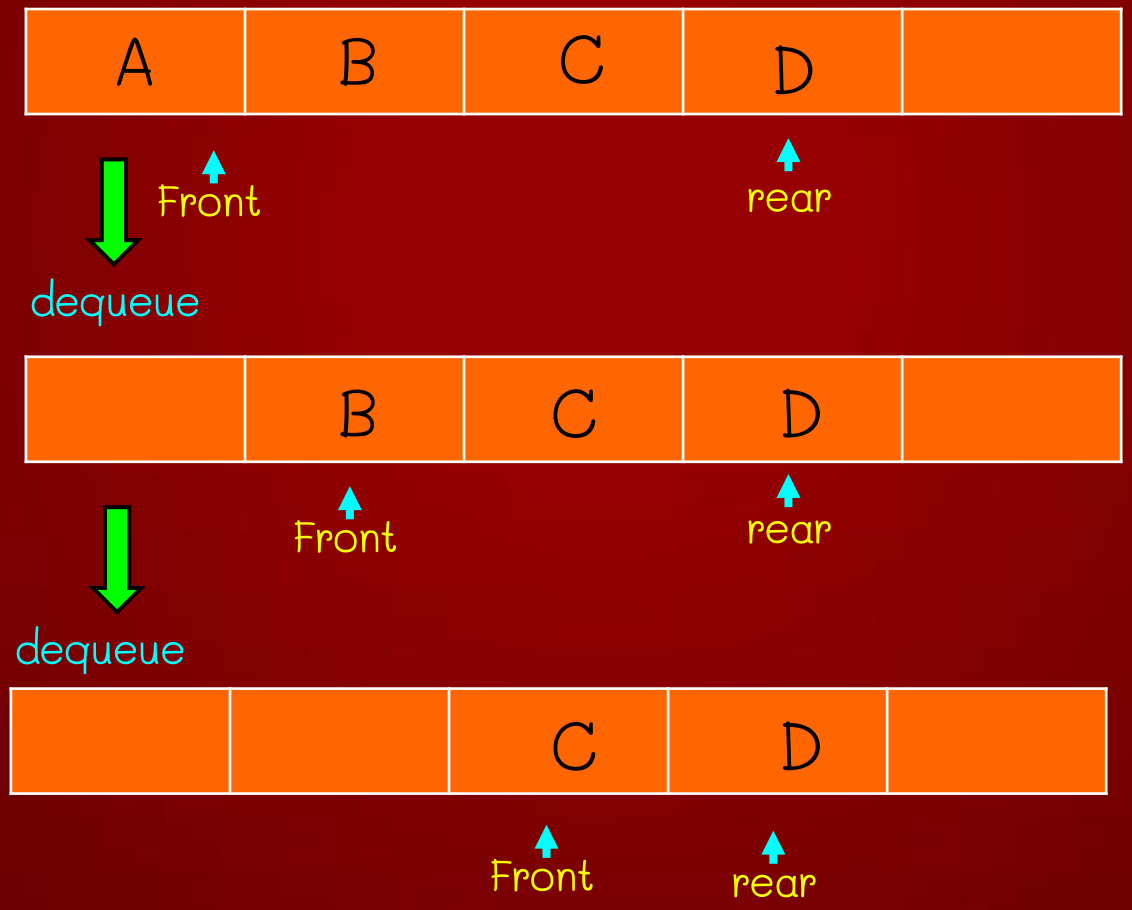
```
DELETE-Q algorithm

If (rear == 0)
        write "Underflow"
else
    if (front==rear)
        front ← 0
        rear ← 0
else
        front ← front + 1
```

ตรวจสอบว่าคิวเต็มหรือไม่

ตรวจสอบว่ามีข้อมูลในคิวเพียงค่าเดียวหรือไม่ ซึ่งถ้า front = rear แสดงว่ามีข้อมูลในคิวเพียงค่าเดียว เมื่อต้องการลบข้อมูลออกจะทำให้ไม่เหลือข้อมูลใด ๆ ในคิว จึงทำการกำหนดให้ค่า front และ rear มีค่าเป็น 0 ซึ่งเป็นการแสดงให้เห็นว่าคิวว่างเปล่า

Queue : Array Implementation

การนำสมาชิกออกจากคิว (dequeue)

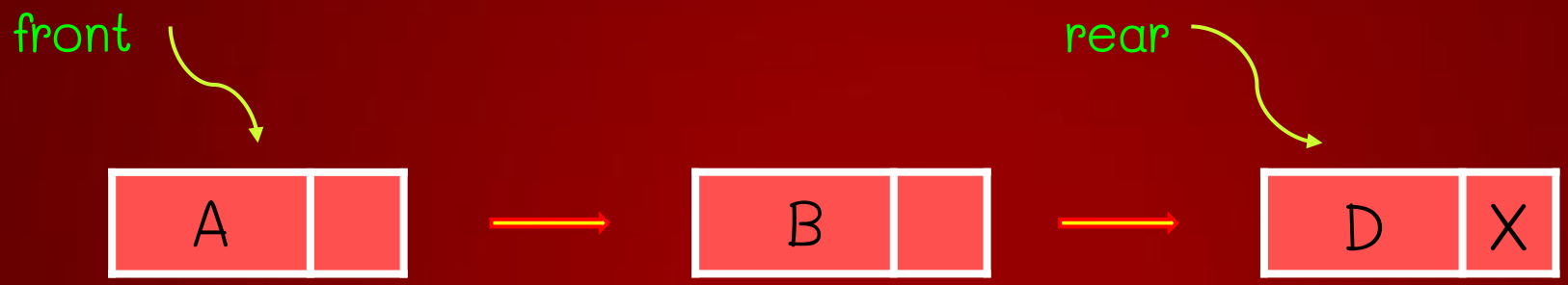


Queue : Array Implementation

การนำสมาชิกออกจากคิว (dequeue) ต่อ

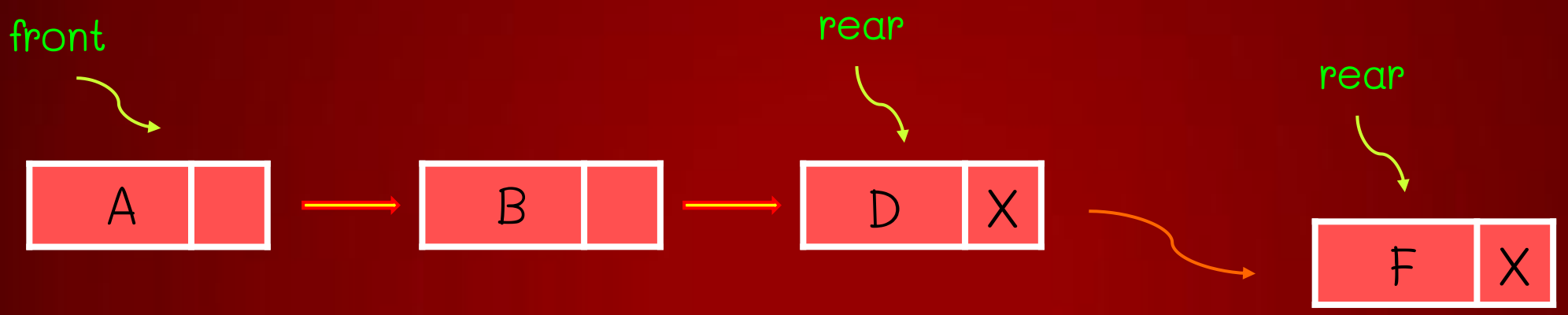
Error : Overflow

Queue : Linked List Implementation



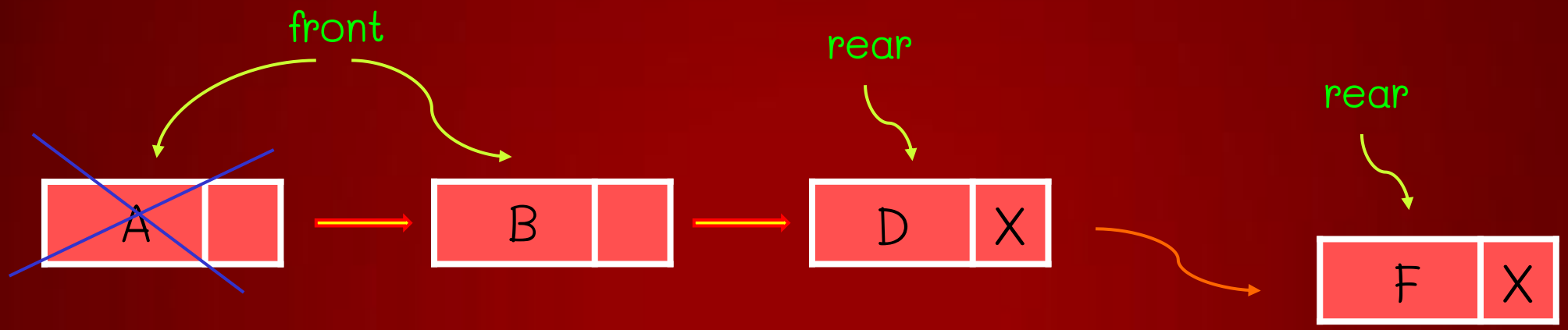

```
struct    node  {
                char              name;
                struct  node      *link;
                }
```

Queue : Linked List Implementation




```
void enqueue (datatype newdata)
{   ...
    rear ⟶ link = newnode;
    rear = newnode;
}
```

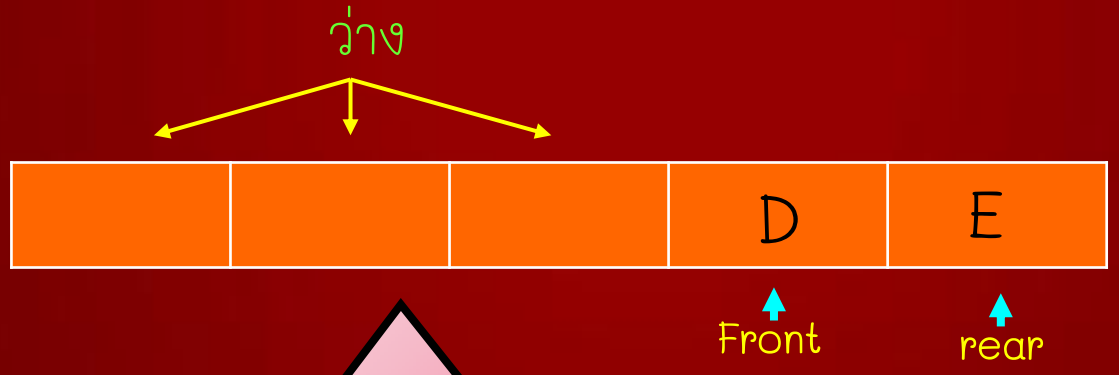
Queue : Linked List Implementation




```
        char dequeue ()
{   ...
    deldata = frontd ⟶ ata;
    front = front ⟶ link;
... }
```

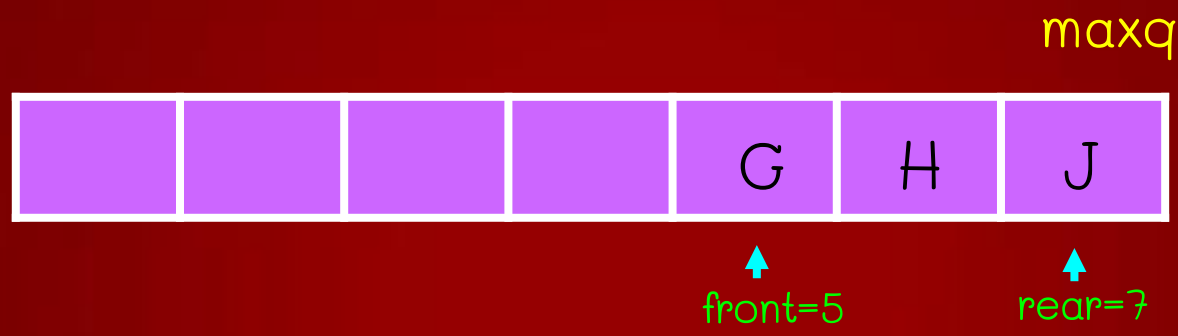
QUEUE

- คิวแบบวงกลม (Circular Queue)
- เป็นการแก้ปัญหาคิวเต็ม ทั้งที่ยังมีเนื้อที่ว่างอยู่
- เป็นการใช้เนื้อที่ให้เกิดประโยชน์สูงสุด

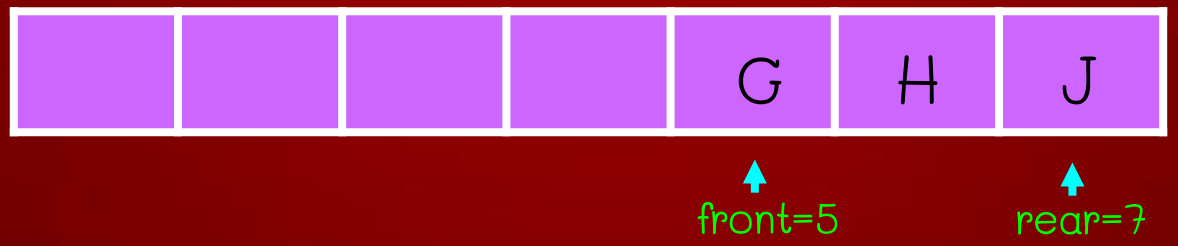


คิวว่าง แต่ไม่สามารถเพิ่มข้อมูลเข้าไปได้อีก เนื่องจาก rear = maxQ

Problem with Array Implementation (size = 7)



Enqueue K



Error :

Overflow

rear=8

Circular Queue

ลักษณะของคิวแบบวงกลม

- เหมือนคิวธรรมดาคือมีตัวชี้ 2 ตัวคือ front และ rear สำหรับแสดงตำแหน่งหัวคิวและท้ายคิวตามลำดับ
- แตกต่างจากคิวธรรมดาคือ คิวธรรมดาเมื่อ rear ชี้อยู่ที่ตำแหน่งสุดท้ายของคิว จะทำให้ไม่สามารถเพิ่มข้อมูลเข้าไปในคิวได้อีก ทั้งที่บางครั้งยังมีที่ว่างเหลืออยู่ก็ตาม

Circular Queue

ลักษณะของคิวแบบวงกลม (ต่อ)

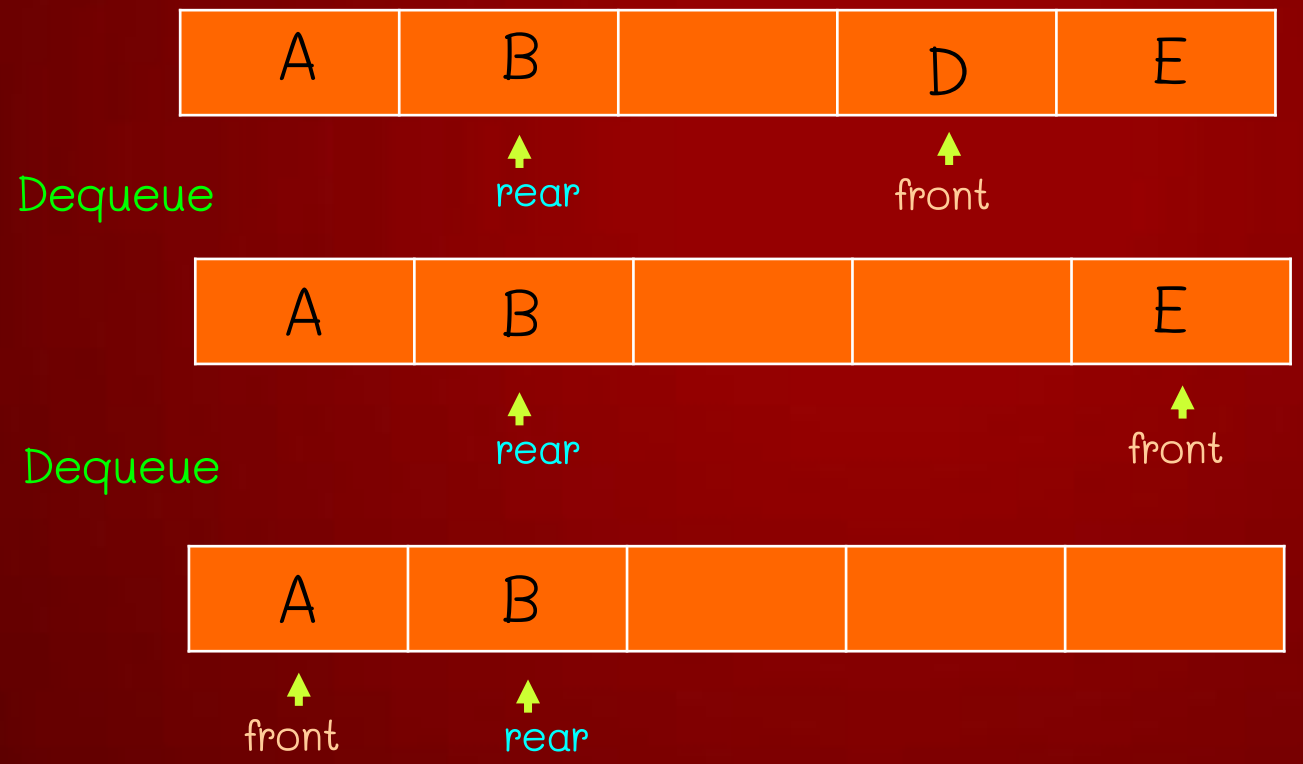
- คิววงกลมจัดการปัญหานี้โดย กรณี rear ชี้อยู่ที่ตำแหน่งสุดท้ายของคิว ถ้าหากมีการเพิ่มข้อมูล ค่าของ rear จะสามารถวนกลับมาชี้ยังตำแหน่งแรกสุดของคิวได้
- ดังนั้นคิววงกลมจะสามารถเพิ่มข้อมูลเข้าไปในคิวได้ จนกว่าคิวจะเต็มจริง ๆ

Circular Queue

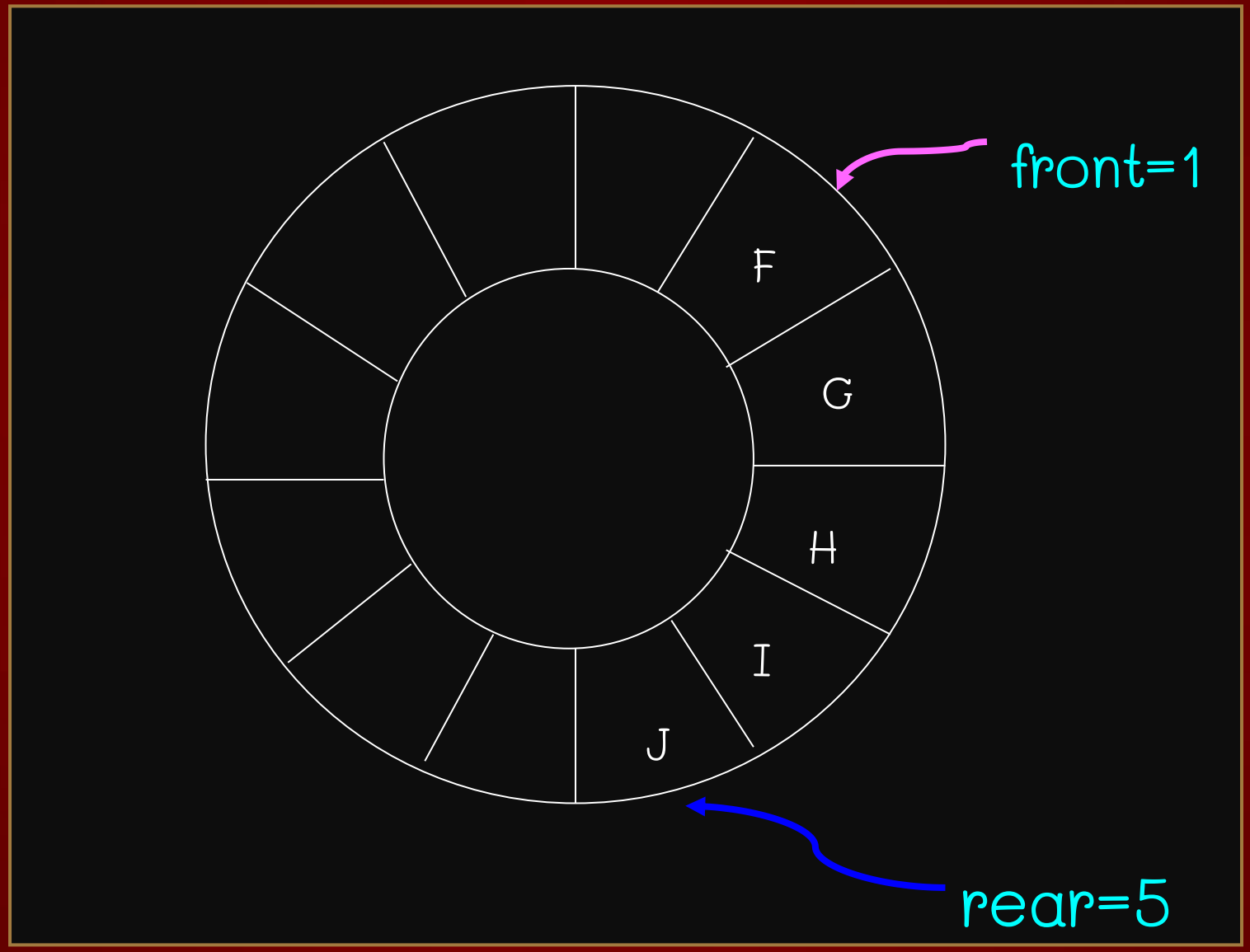
การนำข้อมูลเข้าคิววงกลม

Circular Queue

การนำสมาชิกออกจากคิววงกลม

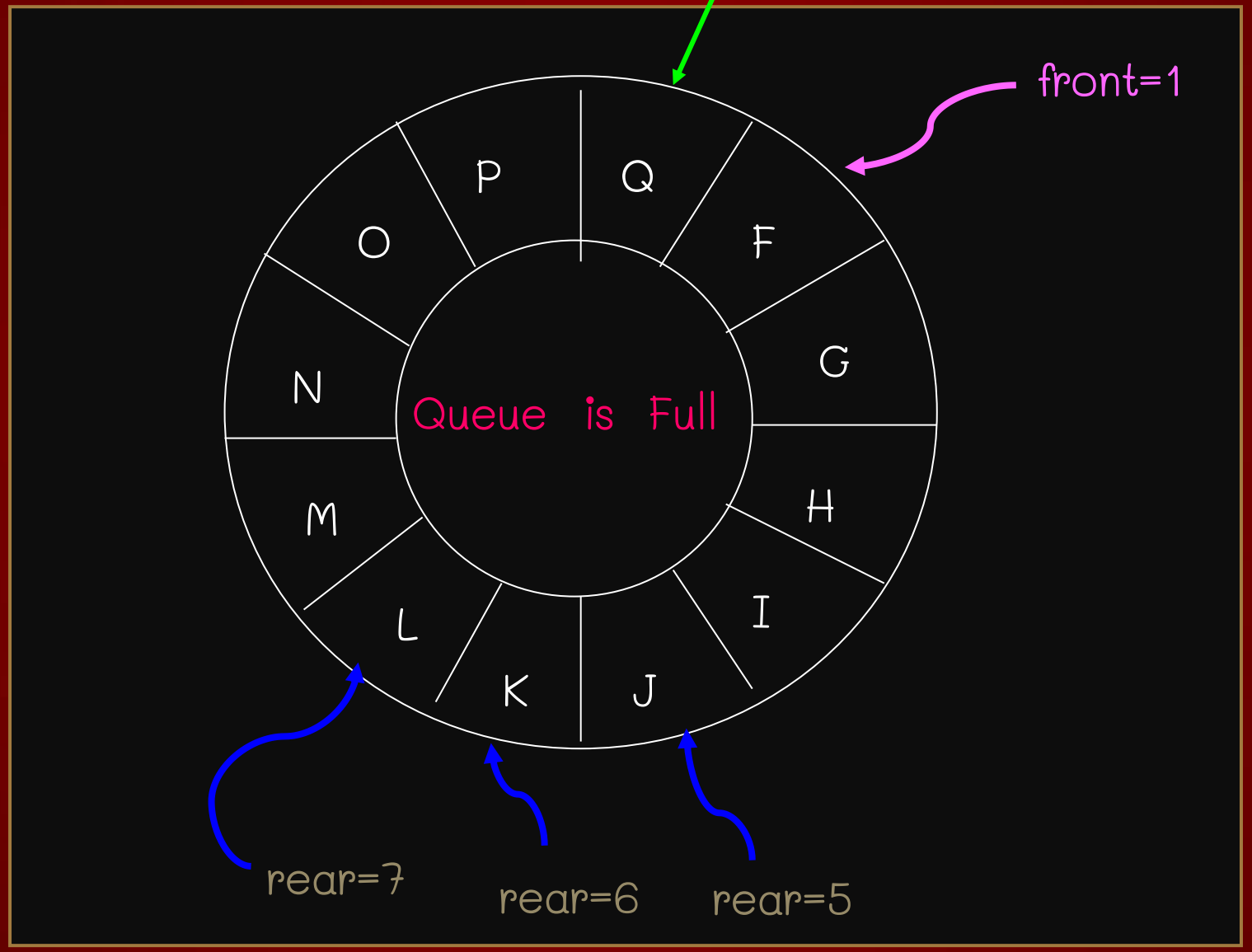


Circular Queue

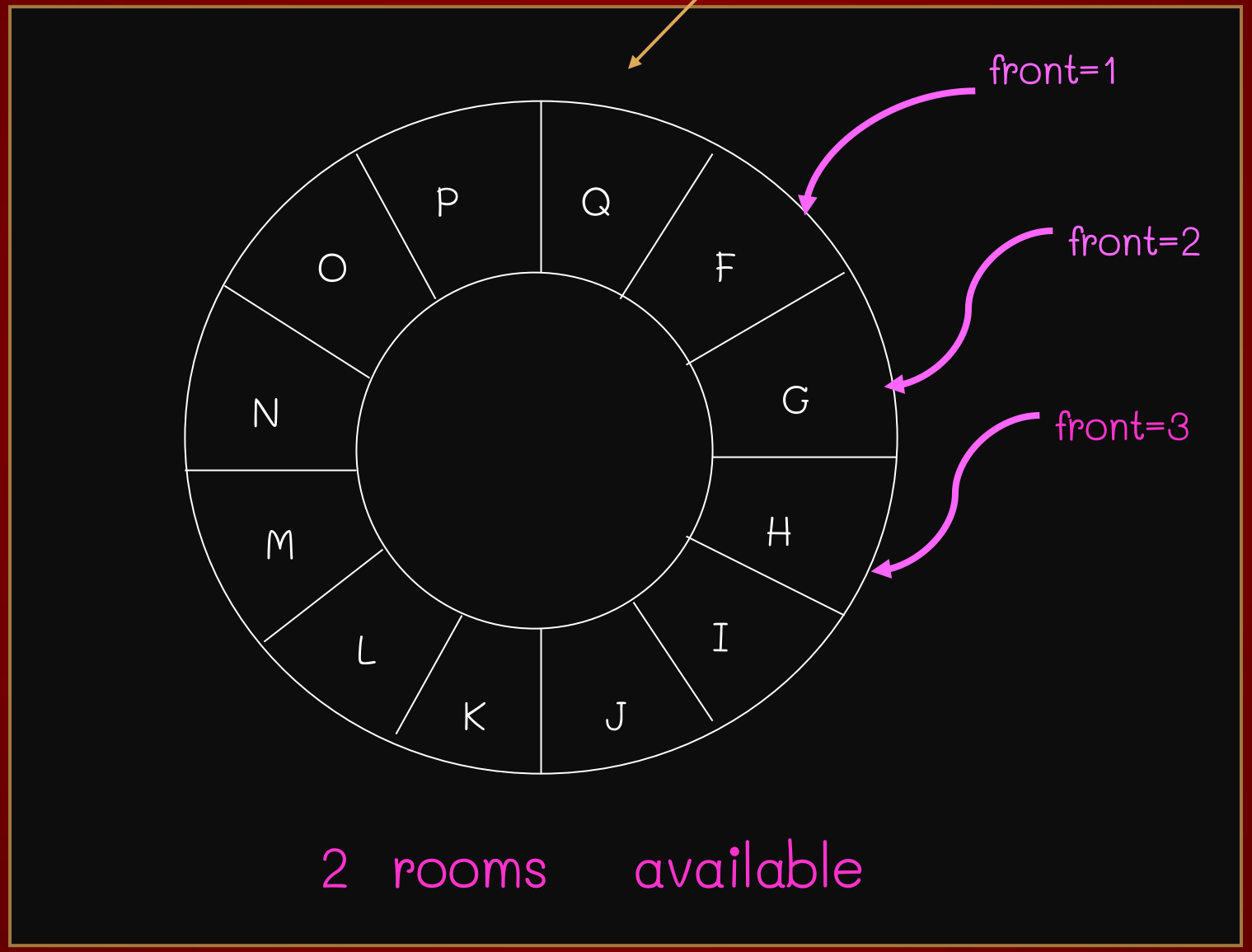


Circular Queue

Enqueue



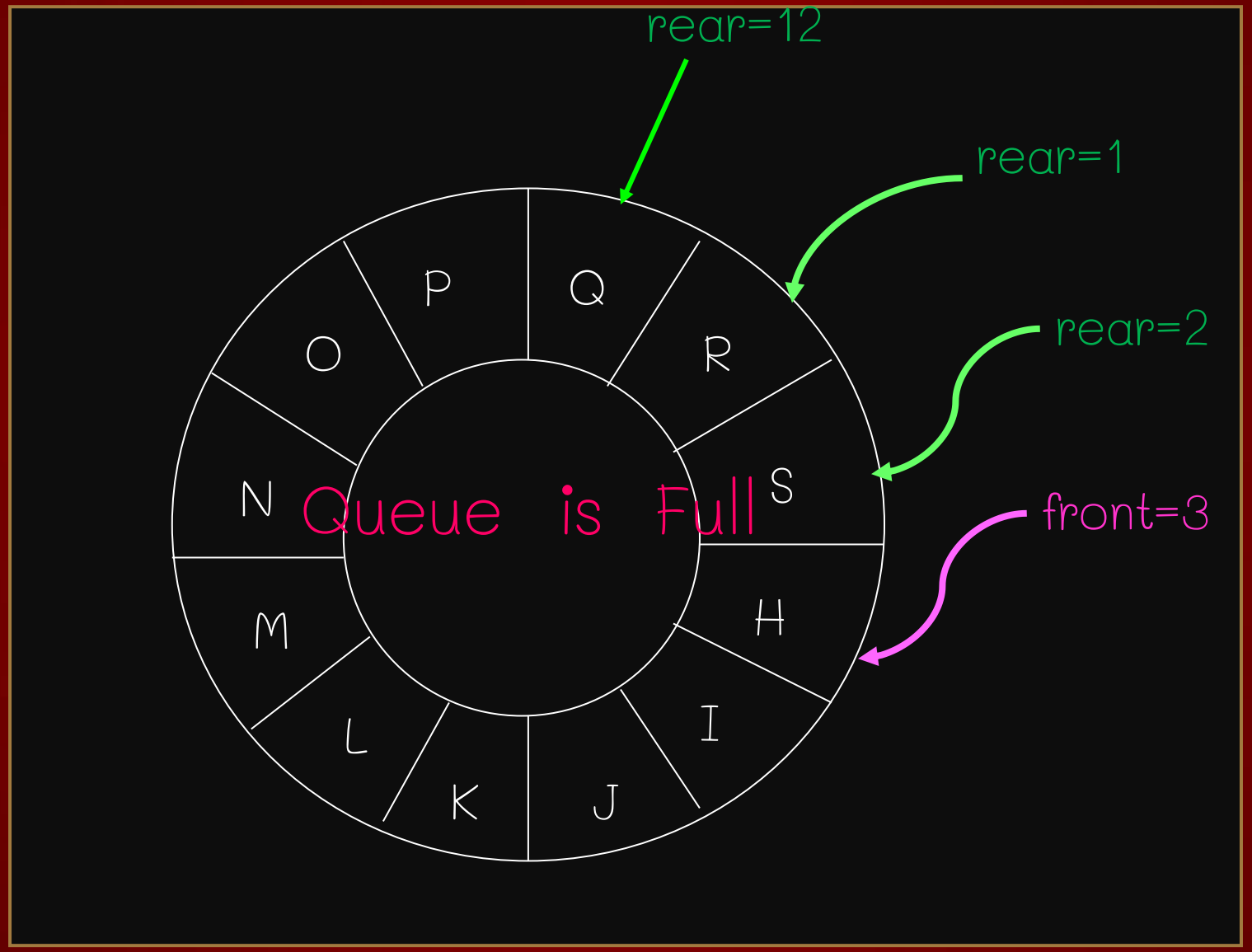
Circular Queue



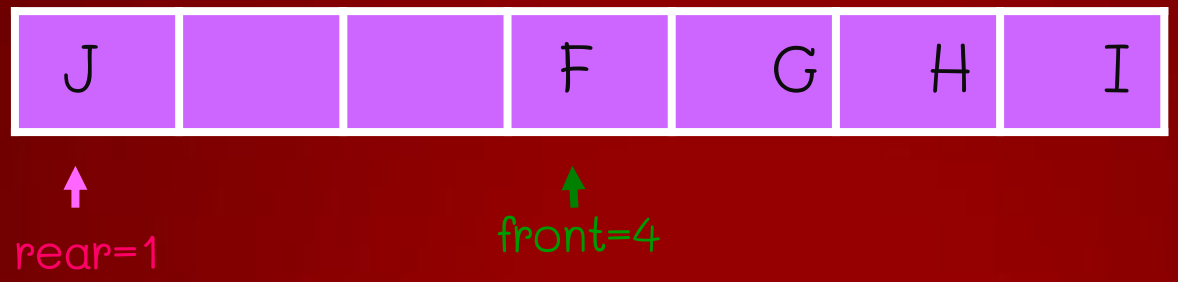
Dequeue

Circular Queue

Enqueue

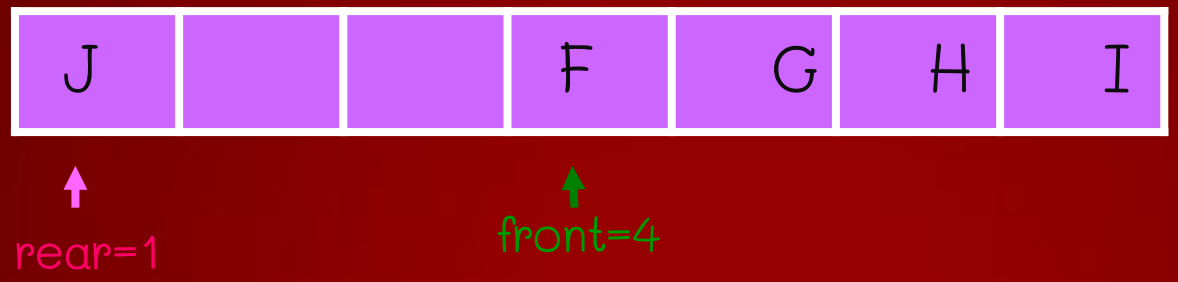


Array Implementation




```
subprogram Enqueue (datatype olddata)
    1. rear ← rear+1
    2. if (front == maxq)
            2.1 front ← 1
    3. return
```

Array Implementation




```
subprogram dequeue (datatype olddata)
    1. front ← front+1
    2. if (front == maxq)
            2.1 front ← 1
    3. return
```

จัดทำโดย

นาย อัษฎาวุธ นิยมญาติ
รหัส 055950201074-2
นาย พงศกร พันธุ์บุตรดี
รหัส 055950201096-5
สาขา วิทยาการคอมพิวเตอร์ (ปวค.59/2)