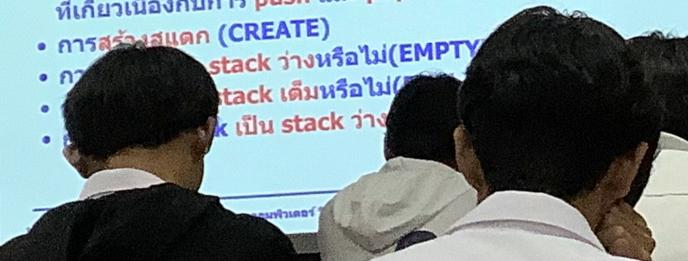


### ກາຮຽກ່າ (Operation) ທີ່ເກີບຂ້ອງກົນໂຄຮງສ້າງຂໍ້ມູນແນນ Stack

- ປົງປັດກິດກາຮຽກ່າຂໍ້ມູນແດກໄດ້ແກ່  
push ດີການປ່າຂໍ້ມູນເກີນໃນສແດກ ແລະ  
pop ດີການປ່າຂໍ້ມູນອອກຈາກສແດກ
- ຢືນທີ່ກ່ອງກົນກາຮຽກ່າຈະກະຫຼາກທ່ານສຸດຂອງ  
ສແດກເສັ່ນ ໂດຍປັດແລ້ວນັກກໍານັດໃນປັດ້ວ້າສຸດຂອງ  
ຂອງສແດກ ເຊິ່ງວ່າ top ສ້າງປົງປັດກິດກາຮຽກ່າ ໃນປົງປັດກິດກາຮຽກ່າ
- ກາຮຽກ່າສແດກ (CREATE)
- ກາຮຽກ່າ stack ວ່າງນີ້ໃໝ່ (EMPTY)
- ກາຮຽກ່າ stack ເຕັມນີ້ໃໝ່ (TOP)
- ປົກປັດກິດກາຮຽກ່າເປັນ stack ວ່າງ

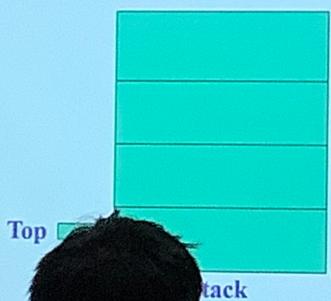


### ຕ້ວເປັນກາຮຽກ່າຂໍ້ມູນເຂົ້າ

1. Top = 0

2. Top = Top + 1

3. Stack[Top] = 10

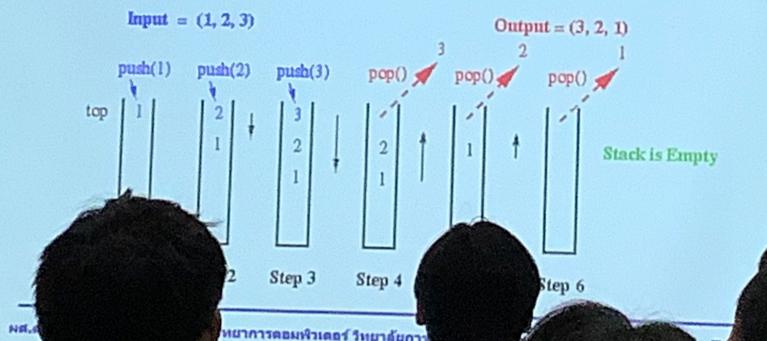


### ກາຮຽກ່າຂໍ້ມູນອອກຈາກຂອງຫຸ້ນ (Pop)

- ກາຮຽກ່າຈະຕຽບກ່າວມີກຳນົດກິດກາຮຽກ່າ
- ຈະດຶງເຂົ້າຂໍ້ມູນທີ່ອຸປະນຸມຸນສຸດອອກມາກ່ອນ ແລ້ວກ່ອນທີ່  
ຈະດຶງຈະມີກາຮຽກ່າສອນວ່າກອງຂ້ອນວ່າງນີ້ໃໝ່
- ດ້ວຍຈະໄນ້ສາມາຄນປ່າຂໍ້ມູນອອກໄດ້  
ແສດງວ່າກອງຂ້ອນວ່າງ (Stack Empty)
- ດ້ວຍຈະໄນ້ວ່າງຈະປ່າເຂົ້າຂໍ້ມູນອອກແລ້ວເລືອນ  
ນີ້ປັບປຸງຕໍ່ແນ່ນກັດລົງໄປ



- ກາຮຽກ່າຂໍ້ມູນເຂົ້າສູ່ສັດ (Push) ກະຫຼາກທ່ານສຸດຂອງສັດ (Top) ຊຶ່ງຕ້ອງມີ  
ກາຮຽກ່າສອນກ່ອນວ່າສັດແດກເດີມຫຼືໄໝ
- ແລະກາຮຽກ່າຂໍ້ມູນອອກຈາກສັດ (Pop) ກະຫຼາກທ່ານສຸດຂອງສັດແດກເປັນກົນ ໂດຍ  
ດົກສອນວ່າມີສາມາຊື່ອຸປະນຸມຸນໃນສັດກົນຫຼືໄໝ (ດົກສອນວ່າສັດແດກກ່າວງແລ້ວຫຼືໄໝ)

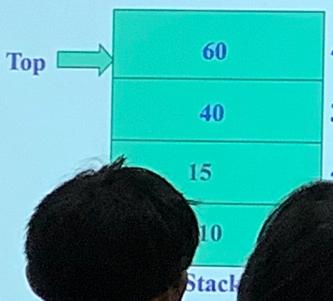


### ຕ້ວເປັນກາຮຽກ່າຂໍ້ມູນເຂົ້າ

1. Top = 0

2. Top = Top + 1

3. Stack[Top] =



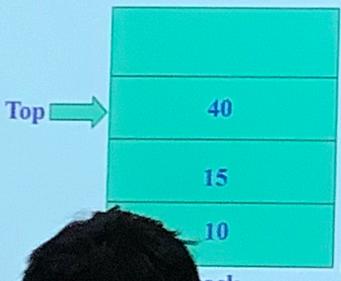
### ຕ້ວເປັນກາຮຽກ່າຂໍ້ມູນອອກ

Temp

60

1. Temp = Stack[Top]

2. Top = Top - 1

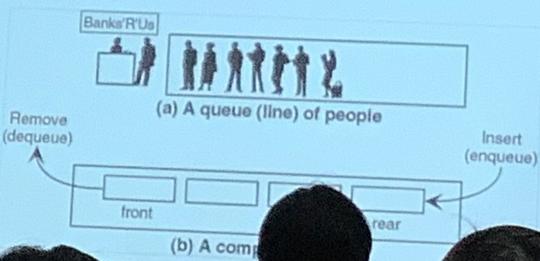
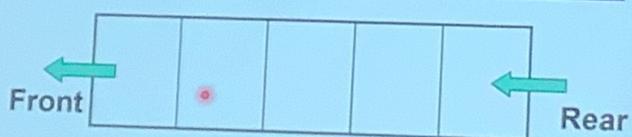


## ການສ້າງ stack ດ້ວຍ Array

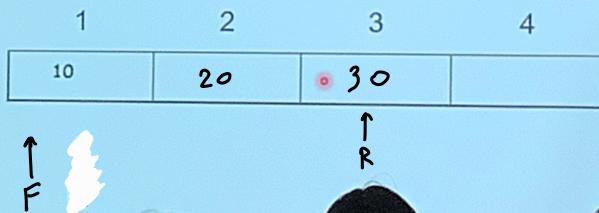
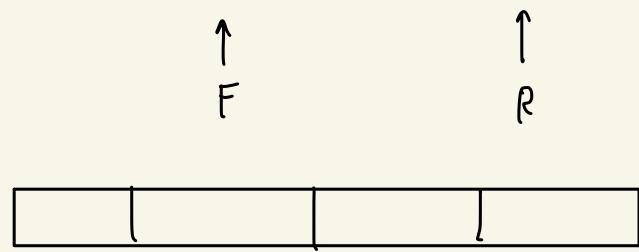
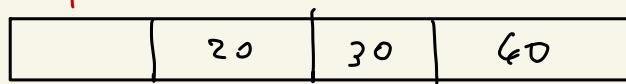
- ໜາຍຄຶ່ງ ການແທນທີ່ຂໍ້ມູນຂອງ stack ດ້ວຍ array ຈຶ່ງເປັນການຈັດສຽບເນື້ອທີ່ ມີການຈັດສຽບເນື້ອທີ່
- ມີການກຳນົດຂາດຂອງ stack ລ່ວງໜ້າວ່າມີຂາດເຫົາໃດ ແລະ ຈະມີການຈັດສຽບເນື້ອທີ່

โครงสร้างข้อมูลแบบคิว (Queues Structure)

- โครงสร้างข้อมูลแบบคิวเป็นโครงสร้างเชิงเส้น
- มีลักษณะของการทำงาน ตรงกันข้ามกับสแตก
- Operations พื้นฐานของ Queue ได้แก่
  - การนำข้อมูลเข้าสู่ Queue เรียกว่า **Enqueue**
  - การนำข้อมูลออกจาก Queue เรียกว่า **Dequeue**
  - การเรียกใช้ข้อมูลจากท้ายแถวของ Queue เรียกว่า **Front**
  - การเรียกใช้ข้อมูลจากท้ายแถวของ Queue เรียกว่า **Rear**
- การสร้าง Queue
  - ใช้ **Array** แทน queue
  - linked list** แทน queue

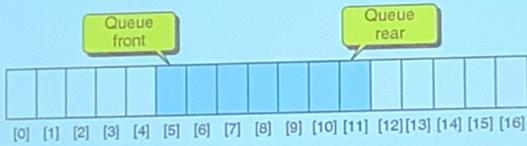
โครงสร้างข้อมูลแบบคิว (Queues Structure)การนำข้อมูลเข้า Enqueueการนำข้อมูลเข้า Enqueue

- การนำข้อมูลใหม่เข้ามาแถวอย่างเพิ่มเข้ามาต้านหลัง
- และจะนำเข้ามาเรื่อยๆ จนเต็ม หรือเรียกว่า แวดคอยเพิ่ม (**Queue Overflow**)

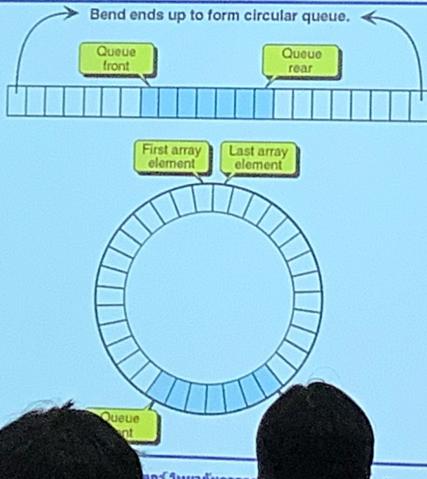
Dequeue

queue underflow

## គິຈວັດກລມ Circular Queue



## គິຈວັດກລມ Circular Queue



## ໂອເປໂຣເຮັດໃນຄົວ

ໂຄຮກສ້າງຂອ້ມແນນດໍາໄວ້ກັບລິສົດໆອື່ນ ຈຶ່ງ  
ຄືອມການພິເນີ້ນຂໍ້ມູນ (insertion) ເຂົາຄົວ  
ແລະມີການຕົກຂໍ້ມູນ (deletion) ຈາກຄົວ

1. ເຝື່ອຄົວວ່າງ (Empty Queue)  
FRONT=0 REAR=0
2. ເຝື່ອຄົວມີສາມາຊີກເດຍ
3. ເຝື່ອຄົວເຕີມ (Full Queue)

## ໂອເປໂຣເຮັດໃນຄົວ

ການພິເນີ້ນຂໍ້ມູນເຂົ້າໄປໃນຄົວ ມີຂັ້ນຕົວ ດັ່ງນີ້

1. ຕຽບສອບວ່າຄົວເຕີມຫຍຼວຍໄວ້ນີ້ ?  
ຕ້ອງເຫັນວ່າ 'Queue Overflow'  
ອອກຈາກກະບວນຄໍາສັ່ງນີ້
2. ເລືອນຕັ້ງທີ່ທ້າຍຄົວໄປອີກ 1 ດໍາແນ່ນ່ງ
3. ນຳຂໍ້ມູນເຂົ້າໄປເກີນໃນຄົວ
4. ຕຽບສອບຕູ້ວ່າ ຂໍ້ມູນທີ່ນຳເຫົ້າໄປເກີນເປັນຂໍ້ມູນຕົວສຸດທ້າຍໃນຄົວຫຍຼວຍໄວ້ ?  
ຕ້ອງເຫັນວ່າ ກີ່ໄດ້ເລືອນຕັ້ງທີ່ທ້າຍໄປຢູ່ທີ່ 1

## ໂອເປໂຣເຮັດໃນຄົວ

```
Procedure : enqueue(QUEUE, N, FRONT, REAR, ITEM)
{This procedure inserts an element ITEM into a queue.}
1. if REAR = N, then : Print : Queue Overflow, and return
   {Queue already filled ?}
2. Set REAR := REAR + 1
3. Set QUEUE(REAR) := ITEM
   {This insert new element.}
4. if FRONT = 0, then : Set FRONT = 1
5. Return
```

## ໂອເປໂຣເຮັດໃນຄົວ

```
size = 3
data = [None]*size #Initialize the QUEUE
REAR = -1
FRONT = -1

def Enqueue(x):
    global REAR,FRONT

    if REAR >= size - 1:
        print("QUEUE Overflow")
    else:
        REAR=REAR+1
        data[REAR] = x

    if FRONT== -1:
        FRONT=0
```

## ໂອເປໂຣເຮັດໃນຄົວ

```
def Dequeue():
    global REAR,FRONT

    if FRONT == -1:
        print("QUEUE Underflow")
    else:
        Deqdata =data[FRONT]
        data[FRONT] = None
        if FRONT==REAR:
            FRONT=-1
            REAR=-1
        else:
            FRONT=FRONT+1

    return Deqdata
```