2.6 Stack

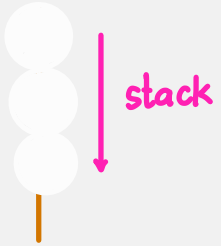
กอง ซ้อน

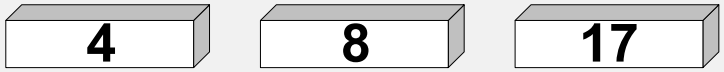
รายการแบบเส้นตรง

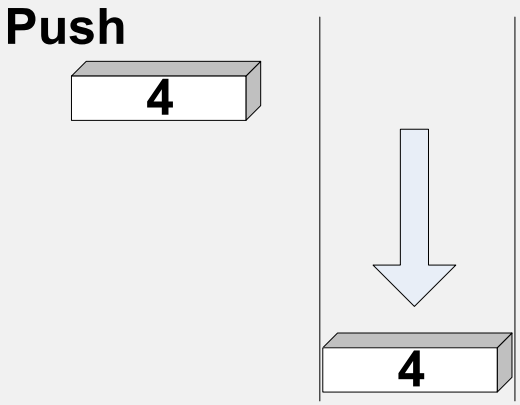
A stack is a linear list with the restriction that insertions and deletions can be performed in only one position, namely, the end of the list, called the top.

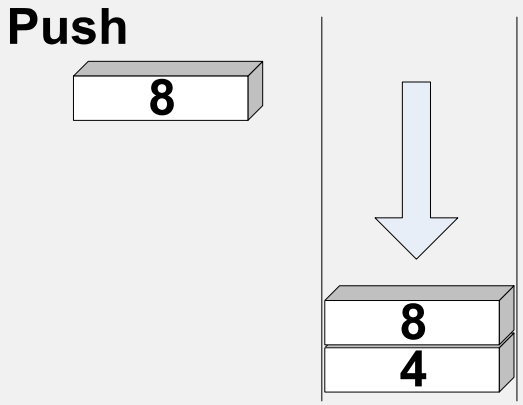
สามารถถูกปฏิบัติ ทำได้ที่นเดียว

Stack is LIFO (last in, first out) data structure

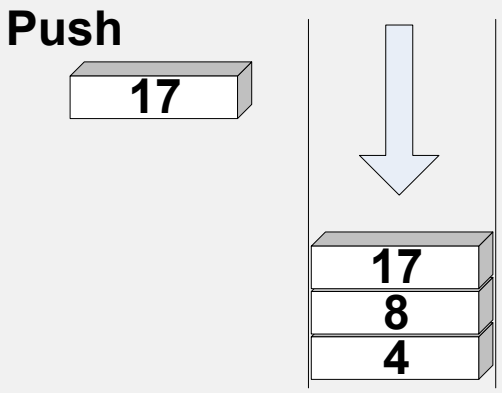




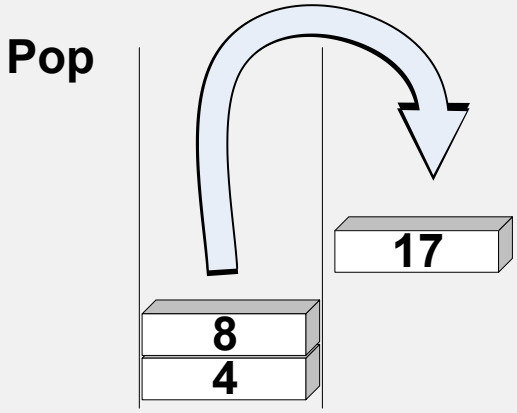




Data เข้าแผนเรียง



นำ Data ออก



The general model is that there is some element that is at the top of the stack, and it is only element that is visible

2.6.2 Implementation of stacks.

- Linked List ยอใปเรื่อยๆ จน RAM เต็ม
- Array ต้อง fix ไว้

```
1. #include <string.h>
2. #include <stdio.h>
3. #include <iostream>
4. using namespace std;
5.  struct Node;
6.  typedef struct Node *Stack;
7. struct Node
8. {     int value;
9.       struct Node *Next;
10. };
```

ประกาศฟังก์ชัน

```
11. int IsEmpty(Stack S);
12. Stack CreateStack(void);
13. void MakeEmpty(Stack S);
14. void Push(int X, Stack S);
15. void Pop(Stack S);
```

8 byte เป็น pointer

12 byte

6 ลงมา 5 ไม่ต้องมีก็ได้

โครง Stack

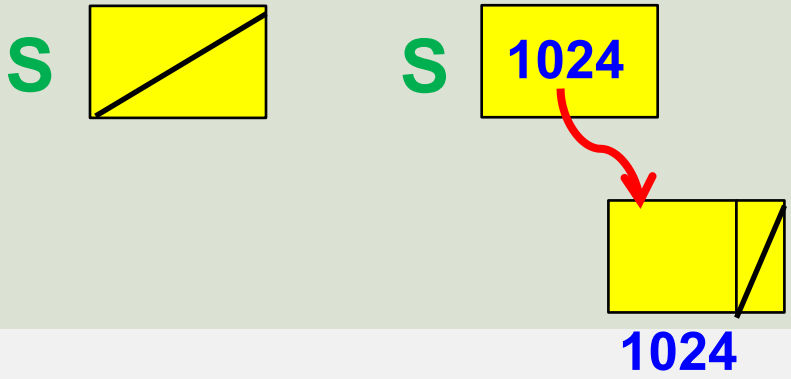
CreateStack สร้างตัวชี้ stack

การ insert จะใช้การ push

การ delete จะใช้การ pop

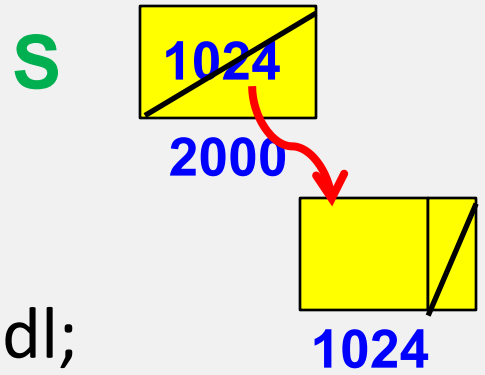
```
int main()
{   Stack S=NULL;
    S=CreateStack();
}
```

8 byte




```
Stack CreateStack(void)
{    Stack S;
     S = new struct Node;
     if (S== NULL)
          cout <<"Out of space!!!"<<endl;
     S->Next=NULL;
     return S;
}
```

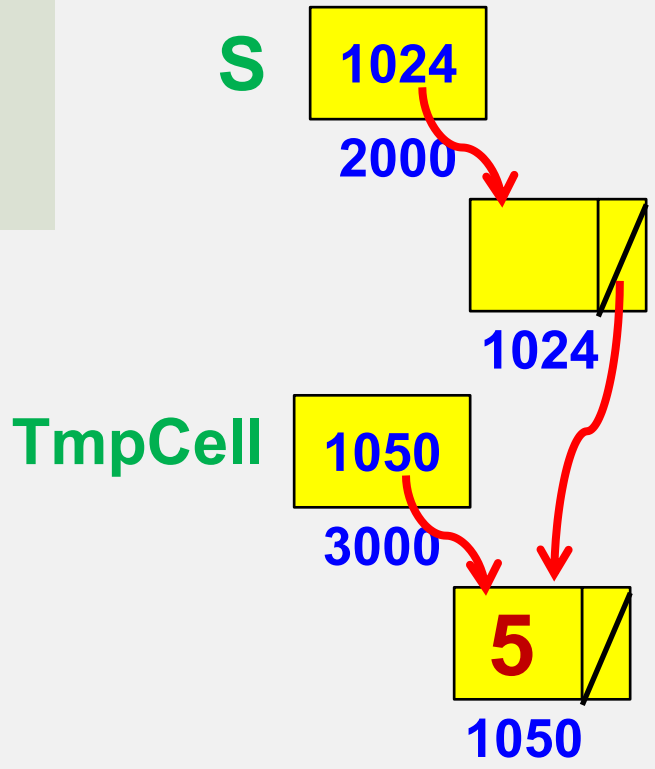
RAM เต็ม



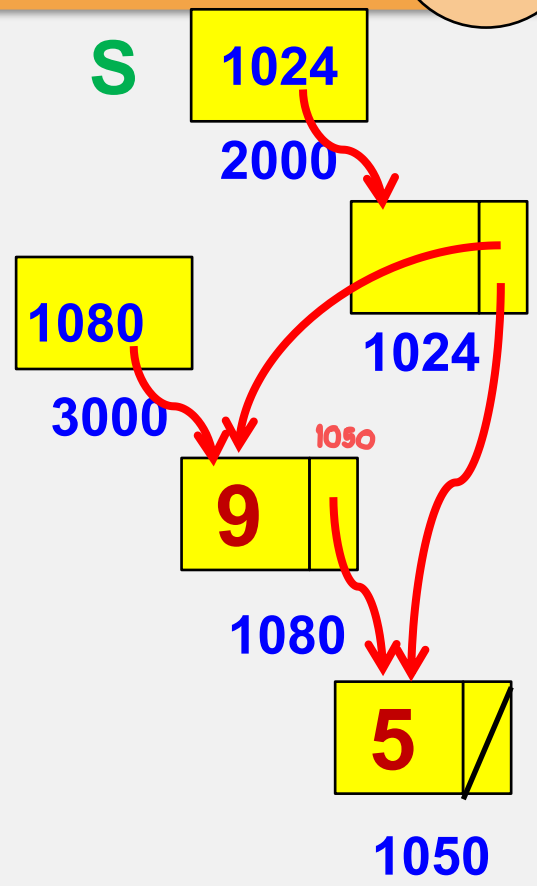
```
int main()
{   Stack S=NULL;
    S=CreateStack();
    Push(5,S);
}
```

S 1024 9000

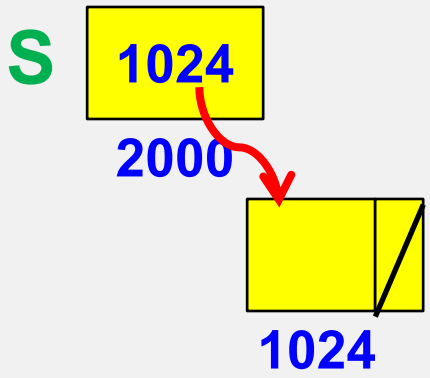
```
void Push(int X,Stack S)
{ 1 Stack TmpCell;
  2 TmpCell = new struct Node;
  3 if(TmpCell == NULL)
  4       cout << "Out of space!!!";
  5 else
  6 {     TmpCell->value = X;
  7       TmpCell->Next = S->Next;
  8       S->Next = TmpCell;
    }
}
```

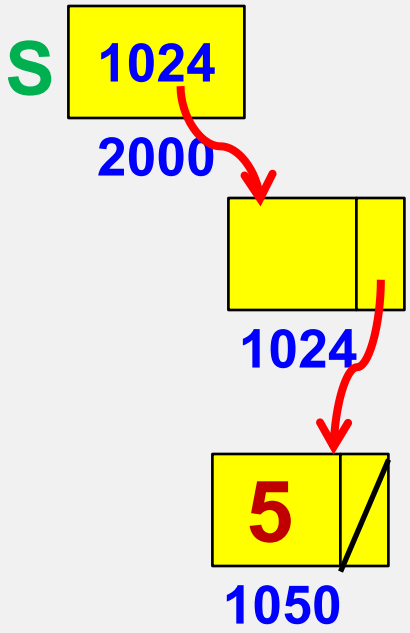


```
void Push(int X,Stack S)
{    Stack TmpCell;
     TmpCell = new struct Node;
     if(TmpCell == NULL)
          cout << "Out of space!!!";
     else
     {    TmpCell->value = X;
          TmpCell->Next = S->Next;
          S->Next = TmpCell;
     }
}
```



```
void MakeEmpty(Stack S)
{    if (S== NULL)
            cout << "Must use CreateStack first" << endl;
     else
       while(!IsEmpty(S))
            Pop(S);
}
```





จริง เป็น 1//

เท็จ เป็น 0//

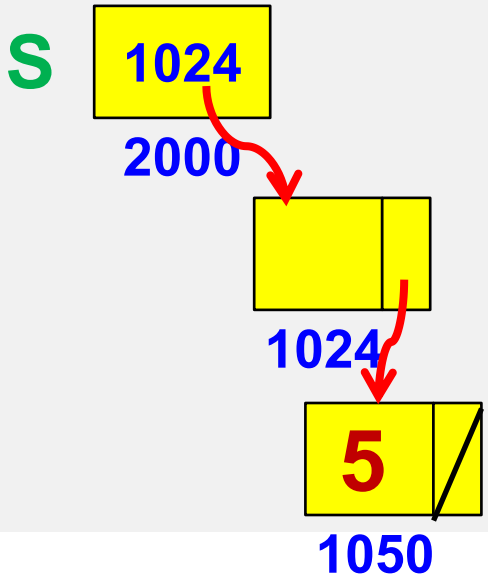
1024

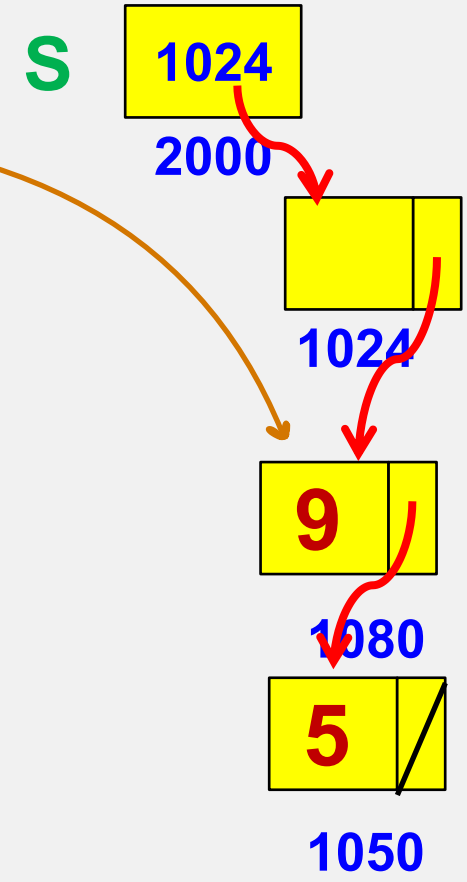
bool

```
int IsEmpty(Stack S)
{   return S->Next == NULL;
}
```

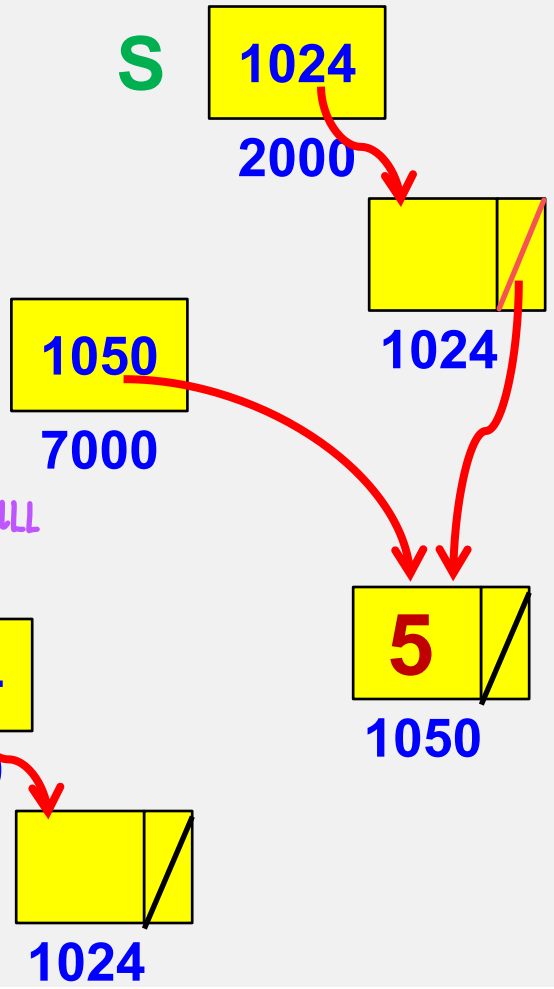
```
int Top(Stack S)
{   if(!IsEmpty(S))
        return S->Next->value;
    cout << "Empty Stack!!!"<< endl;
    return 0;
}
```

เรียก Top ข้อมูลใน stack ไม่หาย

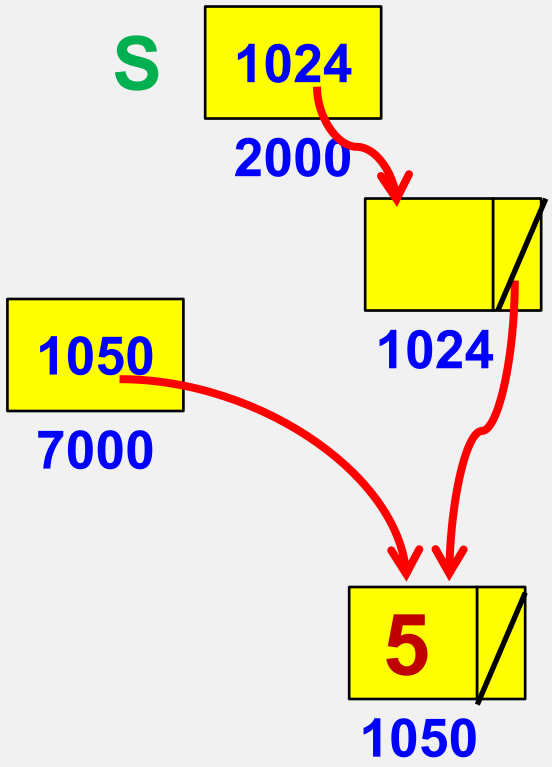




```
void Pop(Stack S)
{  Stack FirstCell;
   if(IsEmpty(S))
         cout << "Empty Stack!!!";
   else
   {     FirstCell = S->Next;
         S->Next = S->Next->Next;
         delete(FirstCell);
   }
}
```



```
void Pop(Stack S)
{  Stack FirstCell;
   if(IsEmpty(S))
         cout << "Empty Stack!!!";
   else
   {     FirstCell = S->Next;
         S->Next = S->Next->Next;
         delete(FirstCell);
   }
}
```



2.6.3 Application

1) Balancing Symbols

```
int main()
{
        cout << “hello”;
}
```

1. Make an empty stack.
2. Read characters until end of file.
3. If the character is an opening symbol, push it onto the stack.
4. If it is a - closing symbol,
   - then if the stack is empty report an error.
   - Otherwise, pop the stack.
5. If the symbol popped is not the corresponding opening symbol, the report an error.
6. At end of file, if the stack not empty report an error.

1. Make an empty stack.
2. Read characters until end of file.
3. If the character is an opening symbol, push it onto the stack.
4. If it is a - closing symbol,
   - then if the stack is empty report an error.
   - Otherwise, pop the stack.
5. If the symbol popped is not the corresponding opening symbol, the report an error.
6. At end of file, if the stack not empty report an error.

2) Infix and Postfix ตอกส่งบ

Infix 4*2

Postfix 42*

4*2 = 42*

4*2+3 = 42*3+

4+2*3 =423*+

4+2+3 =42+3+

4 * 2 + 5 + 6 * 3 =

3) Infix to Postfix Conversion

Operator + , * , (,)

Example

- parentheses

a+ b *c + (d * e + f) *g = a b c * + d e * f + g * +

a*b-c+d

a/b+c*d

a-b*c/d

a-b*c+d

Example

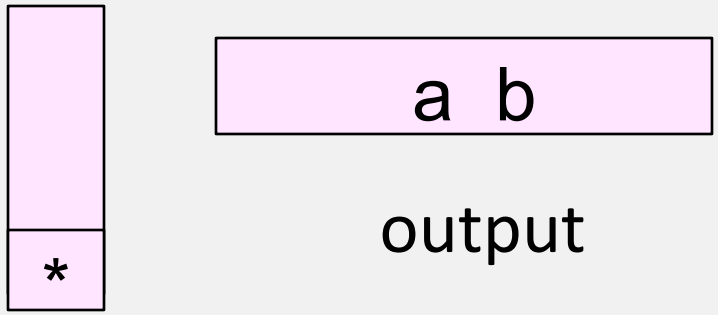
- Stack

a*b-c+d

a/b+c*d

a-b*c/d

a-b*c+d



a + b *c + (d * e + f) *g

a + b * c + (d * e + f) * g

a b c * + d e * f + g * +

เงื่อนไข

1. ถ้า input เป็น operand ให้ print ที่จอภาพ
2. ถ้า input เป็น operator
   2.1 ถ้าเป็น operator ให้เปรียบเทียบ operator ใหม่กับค่าที่อยู่ top ของ stack
   - ถ้าค่าใหม่มี precedence มากกว่า ให้ push ข้อมูลลงใน stack ได้เลย
   - ถ้าค่าใหม่มี precedence **น้อยกว่าหรือเท่ากับ** ให้ pop ข้อมูลมาพิมพ์จนกว่า precedence จะน้อยกว่าค่าใหม่จะน้อยกว่าค่าใน stack หรือ stack empty แล้ว push ค่าใหม่ลงใน stack
   - ถ้าค่าใหม่เป็นวงเล็บเปิด (ให้ push ลง stack ได้เลย และถือว่า precedence มีค่าน้อยที่สุด
   - ถ้าค่าใหม่เป็น วงเล็บปิด) ให้ pop ข้อมูลขึ้นมาพิมพ์จนกว่าจะเจอเครื่องหมาย (

4) Postfix Expressions

42* = 8

42*3+ = 11

423*+ = 10

42+3+ = 9

Infix : 4 * 2 + 5 + 6 * 3

Postfix : 4 2 * 5 + 6 3 * +

= 31

42*

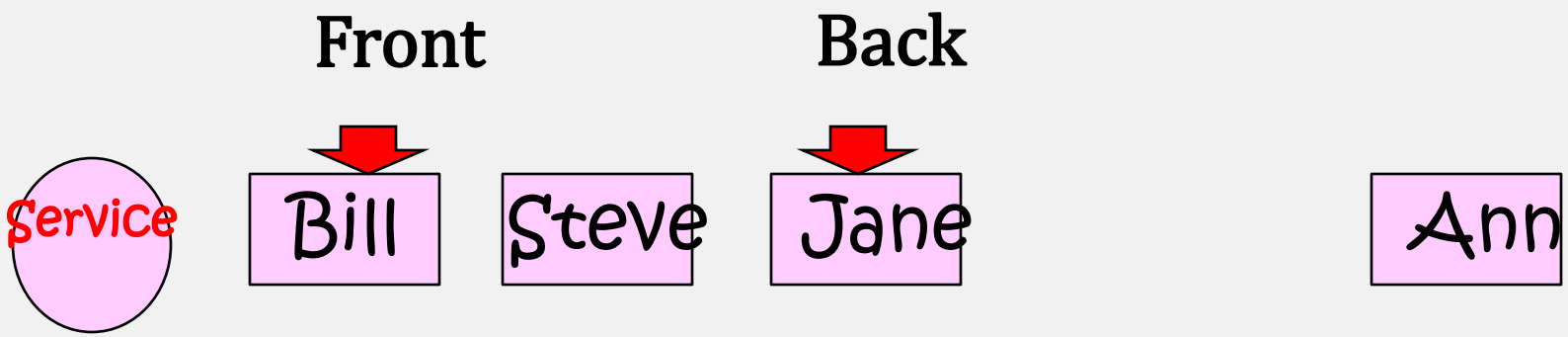
4) Postfix Expressions

Implementation: Stack
Input number : push onto the stack
Input operator : applied to the two numbers that are poped from the stack.

42* = 8
42*3+ = 11
423*+ = 10
42+3+ = 9

Infix : 4 * 2 + 5 + 6 * 3
Postfix : 4 2 * 5 + 6 3 * +
= 31

2.7 Queue are lists. With a queue, however, insertion is done at one end, whereas deletion is performed at the other end.

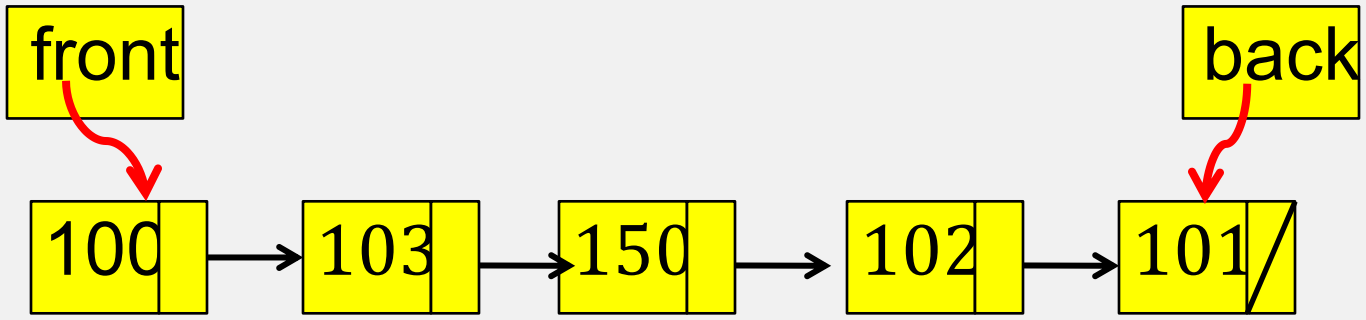


2.7.2 Basic operation

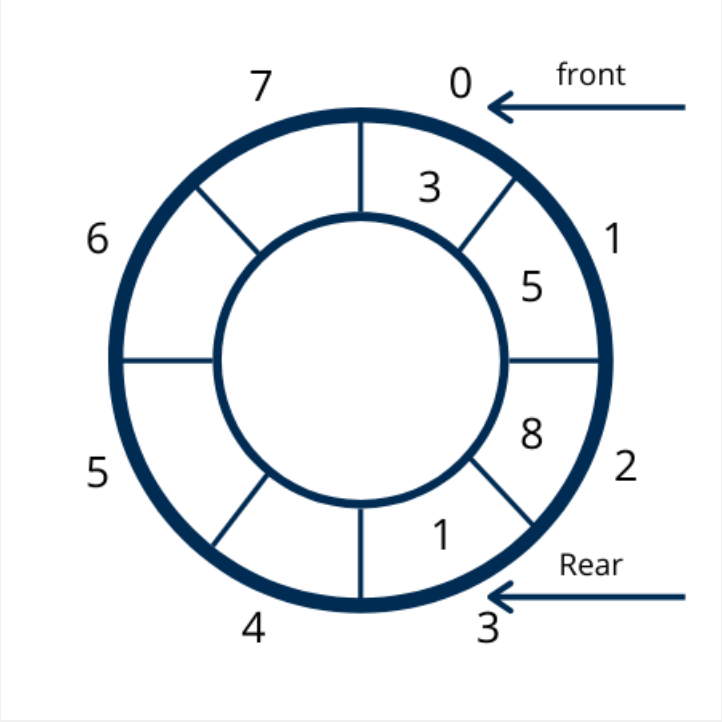
- **Enqueue(x,q)** —Insert item x at the back of queue q.
- **Dequeue(q)** — Return (and remove) the front item from queue q
- **Initialize(q), Full(q), Empty(q)** — Analogous to these operation on stacks.

2.7.2 Implementation of queue.

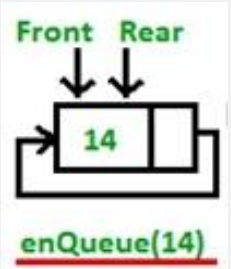
- List (pointer)
- Array

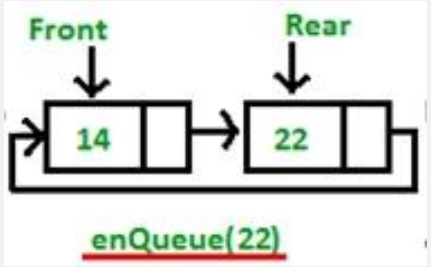


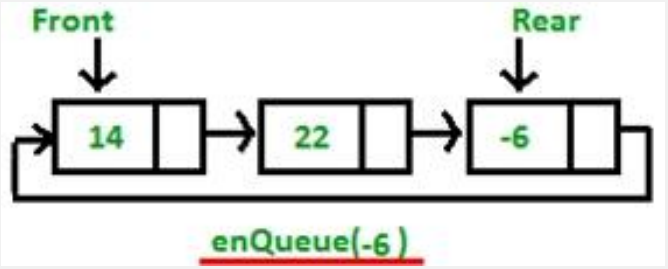
2.7.3 Circular Queue whenever front or back gets to the end of the array. It is wrapped around to the beginning.

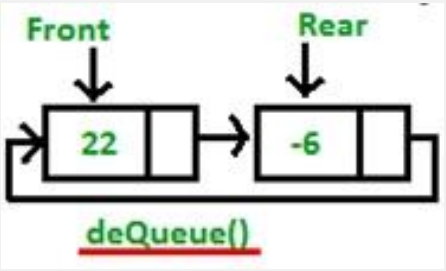


Circular queue : ใช้ Linked list









The josephus problem is the following game:

- N people, numbered 1 to N, are sitting in a circle.
- Starting at person 1, a hot potato is passed.
- After m passed, the person holding the hot potato is eliminated,
- the circle closes ranks,

- and the game continues with the person who was sitting after the eliminate person picking up the hot potato.