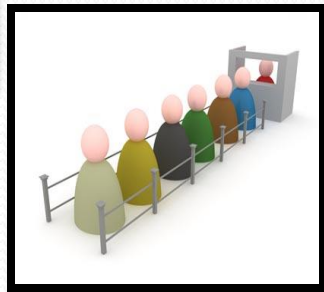




# *Stacks AND Queues*



# Data Structures and Algorithms

# Stacks

- First In Last Out(FILO) or Last In First Out(LIFO) data structure
- Associated with top
- Operations defined:
  - PUSH
  - POP
  - PEEK
- Conditions to be tested:
  - Overflow
  - Underflow



# Defining a Stack structure

```
#define STACKSIZE 100
struct stack
{
    int top;
    int items[STACKSIZE];
};

typedef struct stack STACK;

STACK s;
s.top = -1
```



# Condition: Underflow

When the top is -1

items[99]	
...	
...	
...	
...	
items[3]	
items[2]	
items[1]	
items[0]	
top	-1

S



# Condition: Overflow

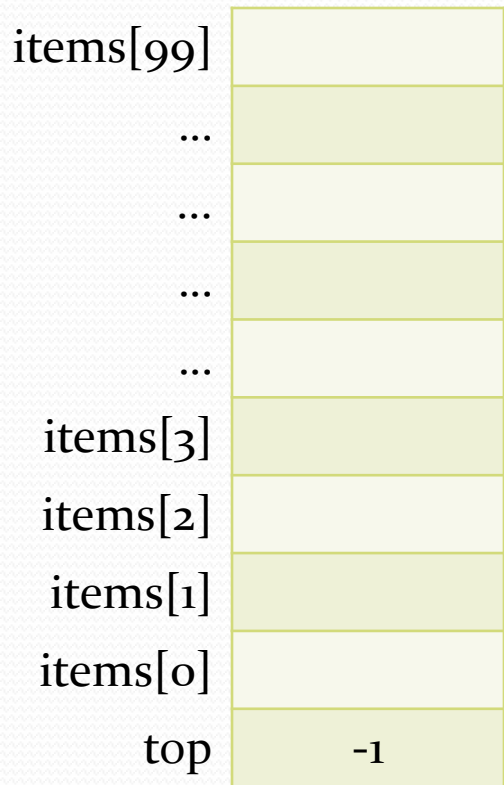
When the top is  
STACKSIZE - 1

items[99]	X <sub>100</sub>
...	...
...	...
...	...
...	...
items[3]	X <sub>4</sub>
items[2]	X <sub>3</sub>
items[1]	X <sub>2</sub>
items[0]	X <sub>1</sub>
top	99

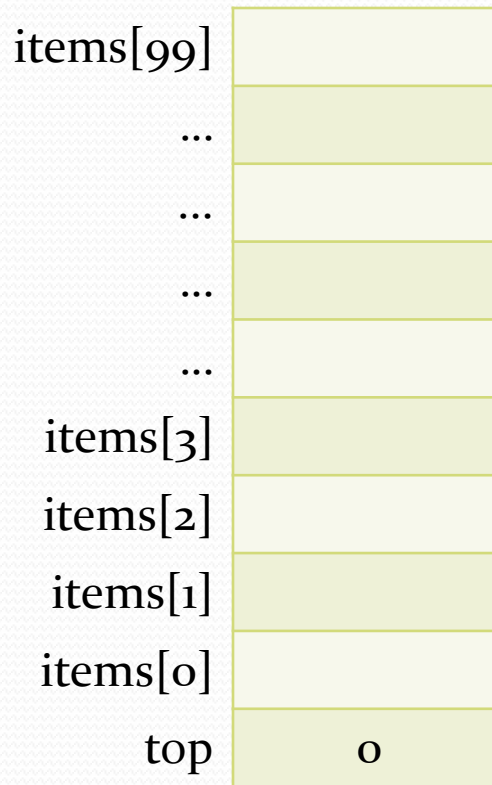
s



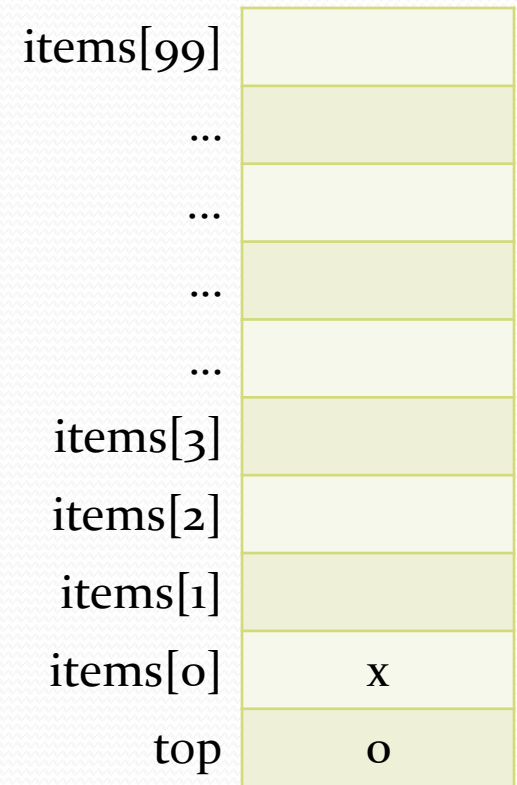
# Operation: PUSH



Initially, s



s.top++



s.items[s.top] = x

The operation is performed after checking overflow condition



# Operation: POP

items[99]	
...	
...	
...	
...	
items[3]	
items[2]	
items[1]	
items[0]	x
top	0

Initially, s

items[99]	
...	
...	
...	
...	
items[3]	
items[2]	
items[1]	
items[0]	x
top	0

$x = s.items[s.top]$

items[99]	
...	
...	
...	
...	
items[3]	
items[2]	
items[1]	
items[0]	
top	-1

s.top--

The operation is performed after checking underflow condition



# Queues

- First In First Out(FIFO) or Last In Last Out(LILO) data structure
- Associated with front and rear
- Operations defined:
  - ENQUEUE
  - DEQUEUE
- Conditions to be tested:
  - Overflow
  - Underflow





# Defining a Queue structure

```
#define MAXQUEUE 100
struct queue
{
    int front;
    int rear;
    int items[MAXQUEUE];
};
```

```
typedef struct queue QUEUE;
```

```
QUEUE q;
q.front = 0;
q.rear = -1;
```

items[99]	
...	
...	
...	
...	
items[2]	
items[1]	
items[0]	
rear	-1
front	0

q



# Condition: Underflow

Operation	Front	Rear
Initial Condition	0	-1
After inserting and Deleting '1' item	1	0
After inserting and Deleting '2' items	2	1
...	...	...
After inserting and Deleting 'n' items	n	n-1

Condition is When the front > rear



# Condition: Overflow

When the rear is  
MAXQUEUE- 1

items[99]	X <sub>100</sub>
...	...
...	...
...	...
...	...
items[2]	X <sub>3</sub>
items[1]	X <sub>2</sub>
items[0]	X <sub>1</sub>
rear	99
front	0 *

q

\* Could be any value between 0 to 99



# Operation: ENQUEUE

items[99]	
...	
...	
...	
...	
items[2]	
items[1]	
items[0]	
rear	-1
front	0

Initially, q

items[99]	
...	
...	
...	
...	
items[2]	
items[1]	
items[0]	
rear	0
front	0

q.rear++

items[99]	
...	
...	
...	
...	
items[2]	
items[1]	
items[0]	x
rear	0
front	0

q.items[q.rear] = x

The operation is performed after checking overflow condition



# Operation: DEQUEUE

items[99]	
...	
...	
...	
...	
items[2]	
items[1]	
items[0]	x
rear	0
front	0

Initially, q

items[99]	
...	
...	
...	
...	
items[2]	
items[1]	
items[0]	x
rear	0
front	0

$x = q.items[q.front]$

items[99]	
...	
...	
...	
...	
items[2]	
items[1]	
items[0]	
rear	0
front	1

$q.front++$

The operation is performed after checking overflow condition



**Thank you.**

