Multiple Stacks and Queues
Multiple Stacks and Queues

- A one dimensional array can be used for building n stacks.
Multiple Stacks and Queues

```c
#define STACK_SIZE 100
#define MAX_STACK 10 // stack number + 1

element stack[STACK_SIZE];
int top[MAX_STACK];
int boundary[MAX_STACK];
int n;  // n < MAX_STACK (number of stacks)
```
Multiple Stacks and Queues

top[0]=boundary[0]=-1;
for(i=1; i<n; i++)
    top[i]=boundary[i]=(STACK_SIZE/n)*i;
boundary[n]=STACK_SIZE-1;
Multiple Stacks and Queues

- If \( \text{top}[i] = \text{boundary}[i+1] \), then the stack \( i \) is completely full.
- If \( \text{top}[i] = \text{boundary}[i] \), then the stack \( i \) is empty.
void push(int i, element item)
{
    if(top[i]==boundary[i+1]) //is there any space in the ith stack?
        stack_full();
    stack[++top[i]]=item;
}
element pop(int i)
{
    if(top[i]==boundary[i] //is the stack empty?
        return stack_empty();
    return stack[top[i]--];
}
Is the stack completely full?  

--Recovery

In order to create some space for a new stack:

1. stack_no < j < n, such that there is free space between stacks j and j+1
   move stacks stack_no+1, stack_no+2, ... j one position right.
   this creates a space between stack_no and stack_no+1.

2. 0 <= j < stack_no, such that there is free space between stacks j and j+1
   move stacks j+1, j+2, ... stack_no one position left.
   this also creates a space between stack_no and stack_no+1.

3. No j satisfying either condition 1 or condition 2, then there is no free space.
References

• Data Structures Notes, Mustafa Ege.