**How to implement stack and queue using arrays. Explain.  
// Stack class**class Stack {   
 **// Array is used to implement stack**constructor()   
{   
this.items = [];   
}   
  
push(element)   
{  **// push element into the items**this.items.push(element);   
}   
pop()   
{  **// return top most element in the stack   
// and removes it from the stack   
// Underflow if stack is empty**if (this.items.length == 0)   
return "Underflow";   
return this.items.pop();   
}   
peek()   
{ **// return the top most element from the stack   
// but does'nt delete it.**return this.items[this.items.length - 1];   
}   
isEmpty()   
{  **// return true if stack is empty**return this.items.length == 0;   
}   
printStack()   
{   
var str = "";   
for (var i = 0; i < this.items.length; i++)   
str += this.items[i] + " ";   
return str;   
}   
  
}   
 **// creating object for stack class**var stack = new Stack();   
 **// testing isEmpty and pop on an empty stack   
  
// returns false**console.log(stack.isEmpty());   
 **// returns Underflow**console.log(stack.pop());   
 **// Adding element to the stack**stack.push(10);   
stack.push(20);   
stack.push(30);   
 **// Printing the stack element   
// prints [10, 20, 30]**console.log(stack.printStack());   
 **// returns 30**console.log(stack.peek());   
 **// returns 30 and remove it from stack**console.log(stack.pop());  **// returns [10, 20]**console.log(stack.printStack());  **Queue  
// Queue class**class Queue   
{  **// Array is used to implement a Queue**constructor()   
{   
this.items = [];   
}  **// Functions to be implemented   
// enqueue function**enqueue(element)   
{   
**// adding element to the queue**this.items.push(element);   
}   
**// dequeue function**dequeue()   
{   
**// removing element from the queue   
// returns underflow when called   
// on empty queue**if(this.isEmpty())   
return "Underflow";   
return this.items.shift();   
}   
**// front function**front()   
{   
**// returns the Front element of   
// the queue without removing it.**if(this.isEmpty())   
return "No elements in Queue";   
return this.items[0];   
}   
**// isEmpty function**isEmpty()   
{   
**// return true if the queue is empty.**return this.items.length == 0;   
}   
**// printQueue function**printQueue()   
{   
var str = "";   
for(var i = 0; i < this.items.length; i++)   
str += this.items[i] +" ";   
return str;   
}   
}   
  
**// creating object for queue class**var queue = new Queue();   
  
 **// Testing dequeue and pop on an empty queue   
// returns Underflow**console.log(queue.dequeue());   
 **// returns true**console.log(queue.isEmpty());   
 **// Adding elements to the queue   
// queue contains [10, 20, 30, 40, 50]**queue.enqueue(10);   
queue.enqueue(20);   
queue.enqueue(30);   
queue.enqueue(40);   
queue.enqueue(50);   
queue.enqueue(60);   
  
**// returns 10**console.log(queue.front());   
 **// removes 10 from the queue   
// queue contains [20, 30, 40, 50, 60]**console.log(queue.dequeue());  **// returns 20**console.log(queue.front());  **// removes 20   
// queue contains [30, 40, 50, 60]**console.log(queue.dequeue());  **// printing the elements of the queue   
// prints [30, 40, 50, 60]**console.log(queue.printQueue());