#Binary Tree

class Node {

constructor(value){

this.value = value

this.left = null

this.right = null

}

}

class BinarySeachTree {

constructor(){

this.root = null

}

insert(value){

var newNode = new Node(value);

if(this.root === null){

this.root = newNode;

return this;

}

let current = this.root;

while(current){

if(value === current.value) return undefined;

if(value < current.value){

if(current.left === null){

current.left = newNode;

return this;

}

current = current.left;

} else {

if(current.right === null){

current.right = newNode;

return this;

}

current = current.right;

}

}

}

}

#Stack

class Stack {

constructor(){

this.data = [];

this.top = 0;

}

push(element) {

this.data[this.top] = element;

this.top = this.top + 1;

}

length() {

return this.top;

}

peek() {

return this.data[this.top-1];

}

isEmpty() {

return this.top === 0;

}

pop() {

if( this.isEmpty() === false ) {

this.top = this.top -1;

return this.data.pop(); // removes the last element

}

}

print() {

var top = this.top - 1; // because top points to index where new element to be inserted

while(top >= 0) { // print upto 0th index

console.log(this.data[top]);

top--;

}

}

reverse() {

this.\_reverse(this.top - 1 );

}

\_reverse(index) {

if(index != 0) {

this.\_reverse(index-1);

}

console.log(this.data[index]);

}

}

#Queue

function Queue() {

this.elements = [];

}

Queue.prototype.enqueue = function (e) {

this.elements.push(e);

};

Queue.prototype.dequeue = function () {

return this.elements.shift();

};

Queue.prototype.isEmpty = function () {

return this.elements.length == 0;

};

Queue.prototype.peek = function () {

return !this.isEmpty() ? this.elements[0] : undefined;

};

Queue.prototype.length = function () {

return this.elements.length;

};

let q = new Queue();

for (let i = 1; i <= 7; i++) {

q.enqueue(i);

}

// get the current item at the front of the queue

console.log(q.peek()); // 1

// get the current length of queue

console.log(q.length()); // 7

// dequeue all elements

while (!q.isEmpty()) {

console.log(q.dequeue());

}