**Data Structure and Algorithm Practicals**

6. Demonstration of Priority Queue

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<script src="queue.js"></script>

<title>Document</title>

</head>

<body>

</body>

</html>

function Queue() {

var items = [];

this.enqueue = function(element){

items.push(element);

}

this.dequeue = function(){

return items.shift();

}

this.front = function(){

return items[0];

}

this.isEmpty = function(){

return items.length == 0;

}

this.clear = function(){

items = [];

}

this.size = function(){

return items.length;

}

this.print = function(){

console.log(items.toString());

}

}

function PriorityQueue() {

var items = [];

function QueueElement(element, priority){

this.element = element;

this.priority = priority;

}

this.enqueue = function(element, priority){

var queueElement = new QueueElement(element, priority);

if (this.isEmpty()){

items.push(queueElement);

} else {

var added = false;

for (var i=0; i<items.length; i++){

if (queueElement.priority < items[i].priority){

items.splice(i, 0, queueElement);

added = true;

break;

}

}

if (!added){

items.push(queueElement);

}

}

}

this.dequeue = function(){

return items.shift();

}

this.front = function(){

return items[0];

}

this.isEmpty = function(){

return items.length == 0;

}

this.clear = function(){

items = [];

}

this.size = function(){

return items.length;

}

this.print = function(){

for (var i=0; i<items.length; i++){

console.log(items[i]);

}

console.log(items);

}

}

var priorityQueue = new PriorityQueue();

priorityQueue.enqueue("John", 2);

priorityQueue.enqueue("Jack", 1);

priorityQueue.enqueue("Camila", 1);

priorityQueue.print();

console.log(priorityQueue.dequeue());

priorityQueue.print();

priorityQueue.print();