**WebTech assignment:**

**JS- Data structures:**

**Navachethan.M – 1NT18IS099**

Html:

<!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">

    <title>Document</title>

</head>

<body>

    <h1>Stack, Queue and Priority queue</h1>

    <script src="js\_ds.js"></script>

</body>

</html>

Js:

// program to implement stack data structure

class Stack {

    constructor() {

        this.items = [];

    }

    // add element to the stack

    add(element) {

        return this.items.push(element);

    }

    // remove element from the stack

    remove() {

        if(this.items.length > 0) {

            return this.items.pop();

        }

    }

    // view the last element

    peek() {

        return this.items[this.items.length - 1];

    }

    // check if the stack is empty

    isEmpty(){

       return this.items.length == 0;

    }

    // the size of the stack

    size(){

        return this.items.length;

    }

    // empty the stack

    clear(){

        this.items = [];

    }

}

console.log('stack implementation');

let stack = new Stack();

stack.add(1);

stack.add(2);

stack.add(4);

stack.add(8);

console.log(stack.items);

console.log(stack.peek());

stack.remove();

console.log(stack.items);

console.log(stack.peek());

console.log(stack.isEmpty());

console.log(stack.size());

stack.clear();

console.log(stack.items);

console.log(stack.isEmpty());

//Queue

class Queue {

    constructor() {

      this.items = {};

      this.headIndex = 0;

      this.tailIndex = 0;

    }

    enqueue(item) {

      this.items[this.tailIndex] = item;

      this.tailIndex++;

    }

    dequeue() {

      const item = this.items[this.headIndex];

      delete this.items[this.headIndex];

      this.headIndex++;

      return item;

    }

    peek() {

      return this.items[this.headIndex];

    }

    get length() {

      return this.tailIndex - this.headIndex;

    }

  }

  console.log("Queue implementation");

  const queue = new Queue();

  queue.enqueue(7);

  queue.enqueue(2);

  queue.enqueue(6);

  queue.enqueue(4);

  console.log(queue.items);

  console.log("pop"+queue.dequeue()); // => 7

  console.log(queue.peek());    // => 2

  console.log(queue.length);    // => 3

  //priority queue

  function PriorityQueueFactory() {

    const items = [];

    return {

      enqueue,

      dequeue,

      front,

      isEmpty,

      size,

      print,

    };

    function createQueueElement(element, priority) {

      return {

        element,

        priority,

      };

    }

    function enqueue(element, priority) {

      const newElement = createQueueElement(element, priority);

      let added = false;

      for (let index = 0; index < items.length; index++) {

        const currentElement = items[index];

        if (newElement.priority < currentElement.priority) {

          items.splice(index, 0, newElement);

          added = true;

          break;

        }

      }

      if (!added) {

        items.push(newElement);

      }

    }

    function dequeue() {

      return items.shift();

    }

    function front() {

      return items[0];

    }

    function isEmpty() {

      return items.length === 0;

    }

    function size() {

      return items.length;

    }

    function print() {

      for(const item of items){

        console.log(`element: ${item.element} - priority: ${item.priority}`)

      }

    }

  }

  var q = PriorityQueueFactory();

  q.enqueue('John', 2);

  q.enqueue('Olivia', 1);

  q.enqueue('Karmen', 3);

  q.enqueue('Oliver', 1);

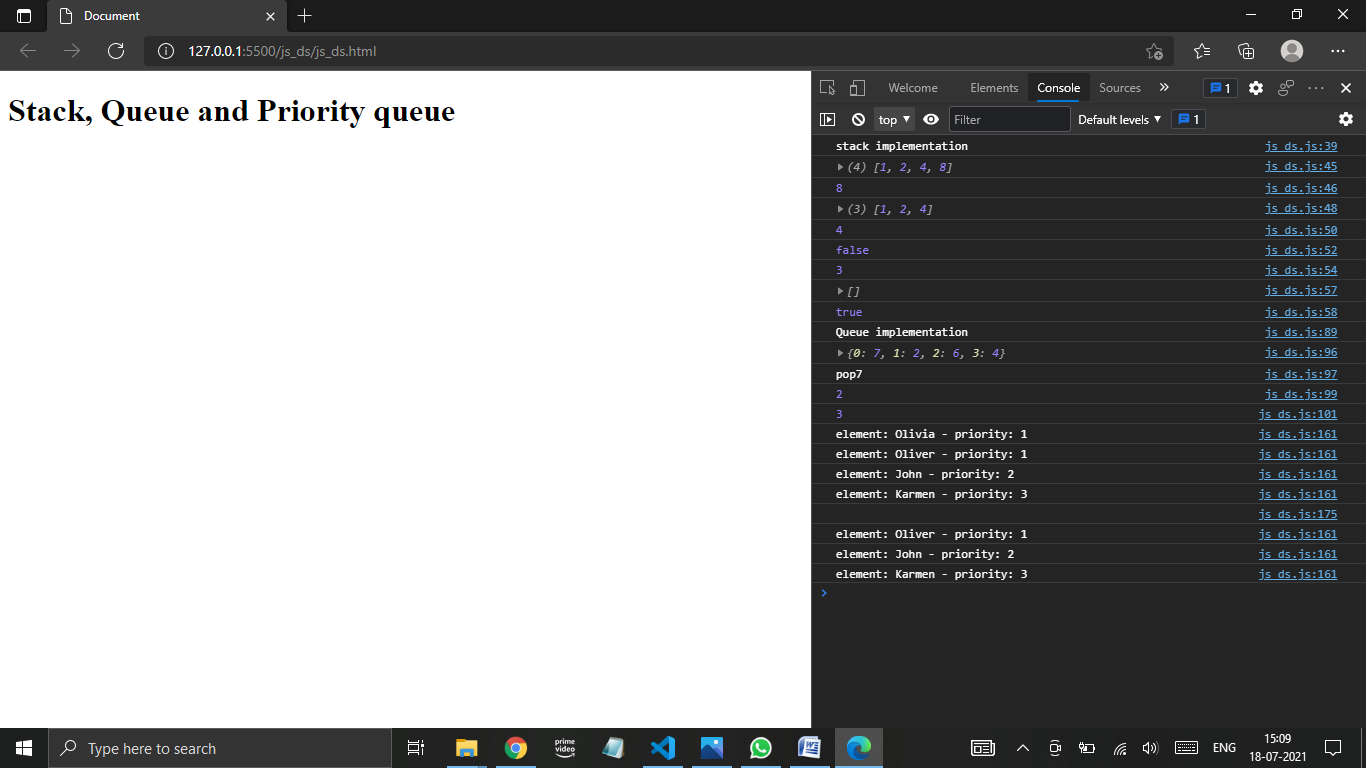
  q.print();

  q.dequeue();

  console.log("");

  q.print();

output:



Linked list:

Html:

<!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">

    <title>Document</title>

</head>

<body>

    <h1>Linked list</h1>

    <script src="js\_linkedlist.js"></script>

</body>

</html>

Js:

class Node {

    // constructor

    constructor(element) {

        this.element = element;

        this.next = null

    }

}

// linkedlist class

class LinkedList {

    constructor() {

        this.head = null;

        this.size = 0;

    }

    // adds an element at the end

    // of list

    add(element) {

        // creates a new node

        var node = new Node(element);

        // to store current node

        var current;

        // if list is Empty add the

        // element and make it head

        if (this.head == null)

            this.head = node;

        else {

            current = this.head;

            // iterate to the end of the

            // list

            while (current.next) {

                current = current.next;

            }

            // add node

            current.next = node;

        }

        this.size++;

    }

    // insert element at the position index

    // of the list

    insertAt(element, index) {

        if (index < 0 || index > this.size)

            return console.log("Please enter a valid index.");

        else {

            // creates a new node

            var node = new Node(element);

            var curr, prev;

            curr = this.head;

            // add the element to the

            // first index

            if (index == 0) {

                node.next = this.head;

                this.head = node;

            } else {

                curr = this.head;

                var it = 0;

                // iterate over the list to find

                // the position to insert

                while (it < index) {

                    it++;

                    prev = curr;

                    curr = curr.next;

                }

                // adding an element

                node.next = curr;

                prev.next = node;

            }

            this.size++;

        }

    }

    // removes an element from the

    // specified location

    removeFrom(index) {

        if (index < 0 || index >= this.size)

            return console.log("Please Enter a valid index");

        else {

            var curr, prev, it = 0;

            curr = this.head;

            prev = curr;

            // deleting first element

            if (index === 0) {

                this.head = curr.next;

            } else {

                // iterate over the list to the

                // position to removce an element

                while (it < index) {

                    it++;

                    prev = curr;

                    curr = curr.next;

                }

                // remove the element

                prev.next = curr.next;

            }

            this.size--;

            // return the remove element

            return curr.element;

        }

    }

    // removes a given element from the

    // list

    removeElement(element) {

        var current = this.head;

        var prev = null;

        // iterate over the list

        while (current != null) {

            // comparing element with current

            // element if found then remove the

            // and return true

            if (current.element === element) {

                if (prev == null) {

                    this.head = current.next;

                } else {

                    prev.next = current.next;

                }

                this.size--;

                return current.element;

            }

            prev = current;

            current = current.next;

        }

        return -1;

    }

    // finds the index of element

    indexOf(element) {

        var count = 0;

        var current = this.head;

        // iterae over the list

        while (current != null) {

            // compare each element of the list

            // with given element

            if (current.element === element)

                return count;

            count++;

            current = current.next;

        }

        // not found

        return -1;

    }

    // checks the list for empty

    isEmpty() {

        return this.size == 0;

    }

    // gives the size of the list

    size\_of\_list() {

        console.log(this.size);

    }

    // prints the list items

    printList() {

        var curr = this.head;

        var str = "";

        while (curr) {

            str += curr.element + " ";

            curr = curr.next;

        }

        console.log(str);

    }

}

// creating an object for the

// Linkedlist class

var ll = new LinkedList();

// testing isEmpty on an empty list

// returns true

console.log(ll.isEmpty());

// adding element to the list

ll.add(10);

// prints 10

ll.printList();

// returns 1

console.log(ll.size\_of\_list());

// adding more elements to the list

ll.add(20);

ll.add(30);

ll.add(40);

ll.add(50);

// returns 10 20 30 40 50

ll.printList();

// prints 50 from the list

console.log("is element removed ?" + ll.removeElement(50));

// prints 10 20 30 40

ll.printList();

// returns 3

console.log("Index of 40 " + ll.indexOf(40));

// insert 60 at second position

// ll contains 10 20 60 30 40

ll.insertAt(60, 2);

ll.printList();

// returns false

console.log("is List Empty ? " + ll.isEmpty());

// remove 3rd element from the list

console.log(ll.removeFrom(3));

// prints 10 20 60 40

ll.printList();

