* Question 1:
  + Create an object student using object literal which has
    - Properties: firstName:String, lastName:String, grades: Array
    - Methods:
      * inputNewGrade(newGrade): push newGrade to grades
      * computeAverageGrade(): return average of grades
    - Create an Array with multiple students which are created using Object.create();
      * Then compute the average grade for all students in the array

const student = {

    firstName: '',

    lastName: '',

    grades: [],

    inputNewGrade: function (newGrade) {

        this.grades.push(newGrade);

    },

    computeAverageGrade() {

        return this.grades.reduce((sum, current, index, array) => sum + current / array.length, 0);

    }

}

const stu1 = Object.create(student);

stu1.firstName = 'John';

stu1.lastName = 'Smith';

stu1.inputNewGrade(88);

stu1.inputNewGrade(98);

stu1.inputNewGrade(86);

stu1.inputNewGrade(80);

const stu2 = Object.create(student);

stu2.firstName = 'John2';

stu2.lastName = 'Smith2';

stu2.inputNewGrade(85);

stu2.inputNewGrade(95);

stu2.inputNewGrade(85);

stu2.inputNewGrade(70);

const students = [stu1, stu2];

const result = students.reduce((average, stu, index, array) => average + stu.computeAverageGrade() / array.length, 0);

console.log(result);

* Question 2: Redo the Question 1 using Constructor Function

function Student(firstName, lastName, grades = []) {

this.firstName = firstName;

this.lastName = lastName;

this.grades = grades;

}

Student.prototype.inputNewGrade = function (newGrade) {

this.grades.push(newGrade);

}

Student.prototype.computeAverageGrade = function () {

return this.grades.reduce((sum, current, index, array) => sum + current / array.length, 0);

}

const stu1 = new Student('John', 'Smith');

stu1.inputNewGrade(88);

stu1.inputNewGrade(98);

stu1.inputNewGrade(86);

stu1.inputNewGrade(80);

const stu2 = new Student('John2', 'Smith2');

stu2.inputNewGrade(85);

stu2.inputNewGrade(95);

stu2.inputNewGrade(85);

stu2.inputNewGrade(70);

const students = [stu1, stu2];

const result = students.reduce((average, stu, index, array) => average + stu.computeAverageGrade() / array.length, 0);

console.log(result);

* Question 3:
  + Add a new method named sort() without parameters in built-in constructor function Array. It’ll sort all elements in the array in ascending order

Array.prototype.mysort = function () {

let arr = this;

let len = arr.length;

for (let i = len - 1; i >= 0; i--) {

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

if (arr[j - 1] > arr[j]) {

let temp = arr[j - 1];

arr[j - 1] = arr[j];

arr[j] = temp;

}

}

}

return arr;

}

console.log([7, 5, 2, 4, 3, 9].mysort());

Question 4: Use Object literal and constructor function to create LinkedList. Methods below:

1) add(value)

2) remove(value)

3) print()

After completion, we can call methods as below and see the results in console if call print().

linkedlist.add(1);

linkedlist.add(2);

linkedlist.add(3);

linkedlist.print(); //in the console, you should see: LinkedList{1,2,3}

linkedlist.remove(3);

linkedlist.print(); //in the console, you should see: LinkedList{1,3}

Object Literal solution:

let linkedlist = {};

linkedlist.add = function(element) {

if (this.value === undefined) {

this.value = element;

this.next = null;

} else {

let current = this;

while (current.next) {

current = current.next;

}

current.next = { value: element, next: null };

}

}

linkedlist.remove = function(element) {

var current = this;

var prev = null;

while (current) {

if (current.value === element) {

if (prev == null) {

this.value = current.next.value;

this.next = current.next.next;

} else {

prev.next = current.next;

}

return true;

}

prev = current;

current = current.next;

}

return false;

}

linkedlist.printHelper = function(list, result) {

if (list.next == null) {

result += list.value;

return result;

}

result += list.value + ',';

return this.printHelper(list.next, result);

}

linkedlist.print = function() {

let result = 'LinkedList{';

result = this.printHelper(this, result);

result += '}';

console.log(result);

}

linkedlist.add(1);

linkedlist.add(2);

linkedlist.add(3);

linkedlist.print(); //in the console, you should see: LinkedList{1,2,3}

linkedlist.remove(3);

linkedlist.print(); //in the console, you should see: LinkedList{1,3}

Constructor Function Solution:

function LinkedList(){

}

LinkedList.prototype.add = function(element) {

if (this.value === undefined) {

this.value = element;

this.next = null;

} else {

let current = this;

while (current.next) {

current = current.next;

}

current.next = { value: element, next: null };

}

}

LinkedList.prototype.remove = function(element) {

var current = this;

var prev = null;

while (current) {

if (current.value === element) {

if (prev == null) {

this.value = current.next.value;

this.next = current.next.next;

} else {

prev.next = current.next;

}

return true;

}

prev = current;

current = current.next;

}

return false;

}

LinkedList.prototype.printHelper = function(list, result) {

if (list.next == null) {

result += list.value;

return result;

}

result += list.value + ',';

return this.printHelper(list.next, result);

}

LinkedList.prototype.print = function() {

let result = 'LinkedList{';

result = this.printHelper(this, result);

result += '}';

console.log(result);

}

let linkedlist = new LinkedList();

linkedlist.add(1);

linkedlist.add(2);

linkedlist.add(3);

linkedlist.print(); //in the console, you should see: LinkedList{1,2,3}

linkedlist.remove(3);

linkedlist.print(); //in the console, you should see: LinkedList{1,3}