Assignment 4

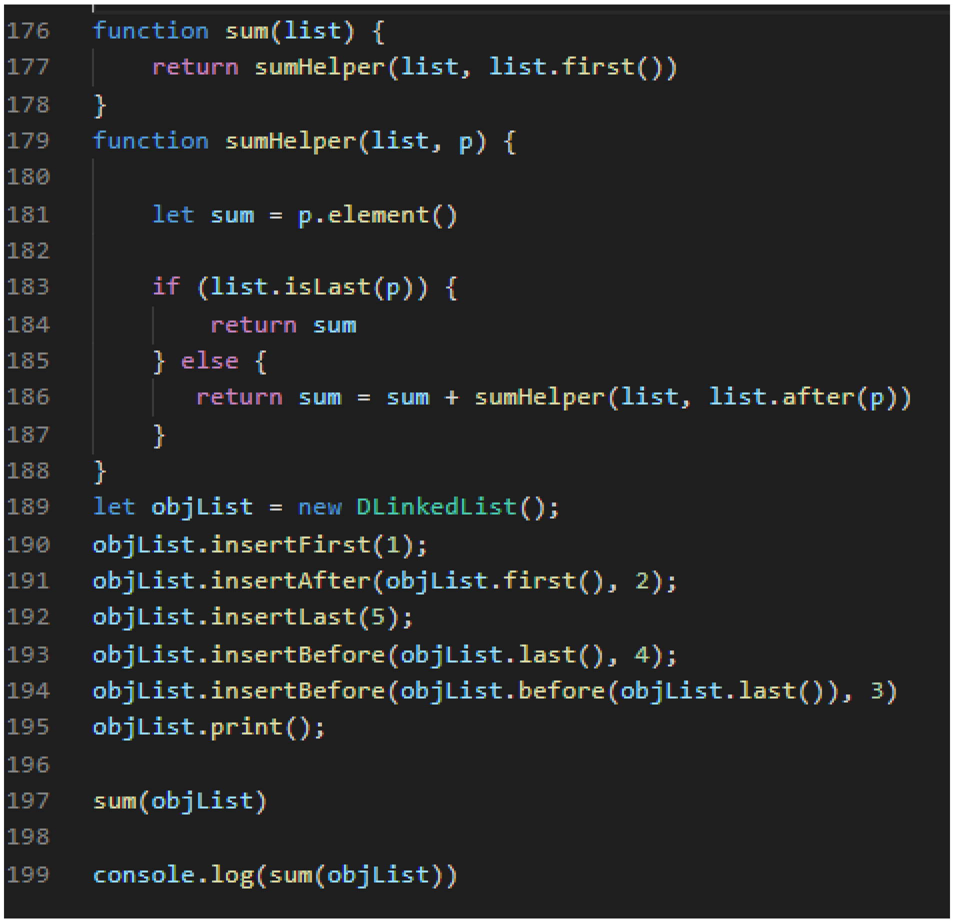
A. (a) Write a recursive method, sum(L), in pseudo-code to calculate the sum of the integers in the list L of integers. First use positions to traverse the list. See the hint in the lecture notes. Analyze line by line your algorithm.

```
Algorithm: sum(L)
    p←L.first                      -------------------O(1)
    return  sumHelper(L, p)--------------------------O(1)

Algorithm:sumHelper(L,p)
    sum←p.element
    if   L.isLast(p) then          --------------------O(1)
        return sum            -----------------------------O(1)
    else
     x ←sumHelper(L, L.after(p))          ------------O(n)
    return sum + x                 ----------------------O(1)

                         so the running time is ------O(n)
```

(b) Write a second recursive algorithm that uses the rank-based operations to traverse the list to calculate the sum. Analyze your algorithm line by line.

```
Algorithm: sum(L)

    return  sumHelper(L, 0) ----------------O(1)

Algorithm:sumHelper(L,r)
sum←L.atRank(r).element
    if  r ≥ L.size()  then--------------------O(1)
        return sum-----------O(1)
    else
     x ←sumHelper(L, r+1) -------------------O(r)

    return x+sum -------------------O(1)

                    --- so the running time is ------O(r)
```

(c) Choose the better algorithm, either (a) or (b), then implement that algorithm in JavaScript using the List.js file provided in a previous assignment



```
function sum(list) {
    return sumHelper(list, list.first())
}
function sumHelper(list, p) {

    let sum = p.element()

    if (list.isLast(p)) {
        return sum
    } else {
       return sum = sum + sumHelper(list, list.after(p))
    }
}
let objList = new DLinkedList();
objList.insertFirst(1);
objList.insertAfter(objList.first(), 2);
objList.insertLast(5);
objList.insertBefore(objList.last(), 4);
objList.insertBefore(objList.before(objList.last()), 3)
objList.print();

sum(objList)

console.log(sum(objList))
```

B. Design a pseudo-code recursive method, findMax(L), that returns the maximum number in the list L. Implement in JavaScript using the List.js file provided previously.

```
Algorithm: findMaximum(L)

        return  findMaximumHelper(L, L.first())--------------------O(1)

Algorithm: findMaximumHelper(L,p)
        if   L.isLast(p) then              ---------------------------------O(1)
            return p.element()             ----------------------------------O(1)
        else
         x ← findMaximumHelper(L, L.after(p)------------------------O(n)
        return max(x, p.element())                   -----------------O(1)

                                           so the running time is ---------O(n)
```