Linked List - Practice Problems

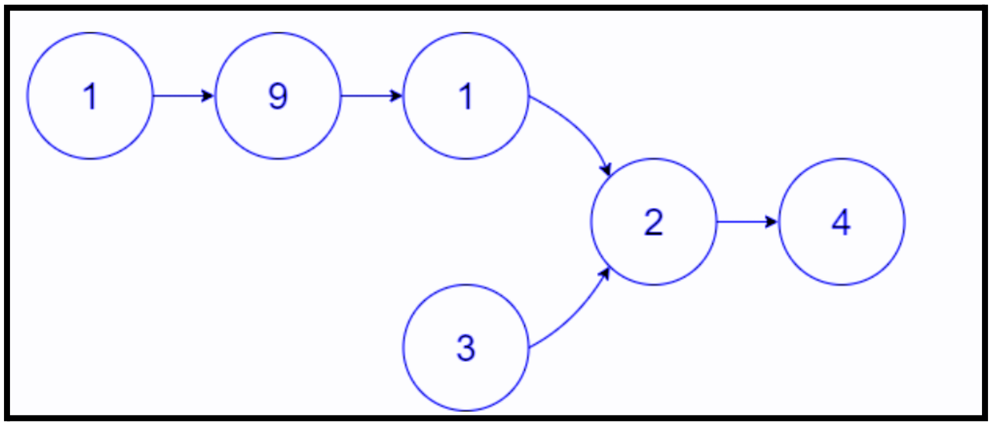
Do the given problems using JavaScript.

1. Write a Program to convert a given array into a linked list. What is the difference between the array and linked list?

   Example:- Input: arr = [1,9,0,7,5]

   Output: 1->9->0->7->5

2. Write a Program to partition a given linked list such that all nodes less than x come before nodes greater than or equal to x, where x is a value inputted by the user along with the new linked list.
3. Write a Program to find the node at which the intersection of two singly linked lists begins.



4. Write a Program to check if a given linked list is a palindrome or not?
5. Write a Program to add an element in the middle of the linked list?
6. Write a Program to delete duplicates in a linked list.
7. Write a Program to reverse a singly linked list.

Solutions

1.

```
class Node {
        constructor() {
          this.data = 0;
          this.next = null;
        }
     }
      var head;
      function insert(root, item) {
        var temp = new Node();
        temp.data = item;
        temp.next = root;
        root = temp;
        return root;
      }
      function print(head) {
        while (head != null) {
          console.log(head.data + " ");
          head = head.next;
        }
      }
      function arrayToList(arr, n) {
        head = null;
        for (var i = n - 1; i >= 0; i--) {
        head = insert(head, arr[i]);
        }
        return head;
      }
      var arr = [2, 1, 332, 41, 25];
      var n = arr.length;
      var head = arrayToList(arr, n);
      print(head);
```

2.

```
function partition(head , x) {
var smallerHead = null, smallerLast = null;
var greaterLast = null, greaterHead = null;
var equalHead = null, equalLast = null;

        while (head != null) {
            if (head.data == x) {
                if (equalHead == null)
                    equalHead = equalLast = head;
                else {
                    equalLast.next = head;
                    equalLast = equalLast.next;
                }
            }
            else if (head.data < x) {
                if (smallerHead == null)
                    smallerLast = smallerHead = head;
                else {
                    smallerLast.next = head;
                    smallerLast = head;
                }
            } else
            {
                if (greaterHead == null)
                    greaterLast = greaterHead = head;
                else {
                    greaterLast.next = head;
                    greaterLast = head;
                }
            }
            head = head.next;
        }

            if (greaterLast != null)
            greaterLast.next = null;

            if (smallerHead == null) {
            if (equalHead == null)
                return greaterHead;
            equalLast.next = greaterHead;
            return equalHead;
        }

            if (equalHead == null) {
            smallerLast.next = greaterHead;
            return smallerHead;
        }

        smallerLast.next = equalHead;
        equalLast.next = greaterHead;
        return smallerHead;
    }
```

3.

```
function(headA, headB) {
    if(headA===null || headB===null) {
        return null;
    }

    let currA = headA;
    let currB = headB;

    while (currA !== currB) {
        currA = currA.next;
        currB = currB.next;
        if(currA === currB) {
            return currA;
        }
        if(currA === null) {
            currA = headB;
        }
        if(currB === null) {
            currB = headA;
        }
    }
    return currA;
}
```

4.

```
function isPalindrome(head) {

        var temp = head;
        var ispalin = true;
        var stack = [];

        while (temp != null) {
            stack.push(temp.data);
            temp = temp.next;
        }

        while (head != null) {
            var i = stack.pop();
            if (head.data == i) {
                ispalin = true;
            } else {
                ispalin = false;
                break;
            }
            head = head.next;
        }
        return ispalin;
    }
```

5.

```
function getCount(node head)
{
    var temp = head;
    var count = 0;
    while (temp!= null)
    {
        count++;
        temp = temp.next;
    }
    return count;
}
function insertMiddle(node head,node n)
{
    var count = getCount(head);
    count/=2;
    var temp = head;
    while (count!= 0)
    {
         count-;
         temp = temp.next;
    }
   n.next = temp.next;
   temp.next = n;
}
```

6.

```
function deleteDuplicate(list) {
    var track = {}; // map to track duplicates
    var temp = list.head;
    var prev = null;
    while (temp) {
        if (track[temp.data]) {
            prev.next = temp.next;
        } else {
            track[temp.data] = true;
            prev = temp;
        }
        temp = temp.next;
    }
    console.log(temp);
}
```

7.

```
function reverseSingleLinkedList(list){
var node = list.head;
var prev = null;
while (node){
   var temp = node.next;
   node.next = prev;
   prev = node;
      if(!temp)
      break;
   node = temp;}
return node;
}
```