Algorithm (L , k)

Input: L- Linked list and k - index to return value

Output: value of the index

If k<0 or k>size of L then

return undefined

Current 🡨the first node of L

Count🡨1

While count < k do

Current🡨node after current in L

Count🡨count+1

Return current node

*function* findKth(*l*, *k*) {

  if (k < 0 || k > l.size()) return undefined;

*let* count = 1;

*let* current = l.first();

  while (count < k) {

    current = l.after(current);

    count++;

  }

  return current.element();

}

2.

Algorithm removeMiddle (L)

Index🡨undefined(to be defined later)

If ( |L| is even) then

Index🡨|L| /2 - 1

Otherwise (if odd)

Index🡨lower Integer of |L| / 2

Current🡨L – first node

countIndex🡨 0

While countIndex< index do

Current🡨node after current in L

CountIndex🡨countIndex+1

Remove current node

Return removed node

*function* removeMiddle(*l*) {

*let* index;

  if (l.size() % 2 === 0) {

    index = l.size() / 2 - 1;

  } else {

    index = Math.floor(l.size() / 2);

  }

*let* current = l.first();

*let* countIndex = 0;

  while (countIndex < index) {

    current = l.after(current);

    countIndex++;

  }

*let* removed = l.remove(current);

  return removed;

}