**SUBMITTED BY: HANIA IMTISAL**

**EX1: DIVIDE TWO INTEGERS**

/\*\*

\* @param {number} dividend

\* @param {number} divisor

\* @return {number}

\*/

/\*

SUBMITTED BY: Hania Imtisal

Approach: Increase the divisor by shifting left (multiplying by 2) until it exceeds the dividend and keep track of how many times it fits the divident which will give the potential result

\*/

var divide = function(dividend, divisor) {

const sign = Math.sign(divisor) !== Math.sign(dividend);

dividend = Math.abs(dividend)

divisor = Math.abs(divisor)

let ret = 0

while (divisor <= dividend) {

let value = divisor

let multiple = 1

while (value + value <= dividend) { //double the value until it is less than the divident

value += value

multiple += multiple

}

dividend = dividend - value

ret += multiple

}

if (ret > ((2\*\*31) - 1)) {

return sign ? -(2\*\*31) : 2\*\*31 - 1

}

return sign ? -ret : ret

};

**EX2: TWO SUM PROBLEM**

/\*\*

\* @param {number[]} numbers

\* @param {number} target

\* @return {number[]}

\*/

/\*

SUBMITTED BY: Hania Imtisal

Approach: use 2 pointers as array is already sorted. Sum 1 and last index, if equal to target return the indexes else if sum is less than target then inc the left to reach the target value and if sum is greater than target, it means we need to eliminate the largest value which will be the last element so dec the right one.

\*/

var twoSum = function(numbers, target) {

var left = 0 ;

var right = numbers.length - 1;

while(left < right)

{

var sum = numbers[left] + numbers[right];

if( sum == target)

{

return[left+1 ,right+1];

}

else if(sum<target)

{

left++;

}

else{

right--;

}

}

};

**EX3: ROTATE LIST**

/\*\*

\* Definition for singly-linked list.

\* function ListNode(val, next) {

\* this.val = (val===undefined ? 0 : val)

\* this.next = (next===undefined ? null : next)

\* }

\*/

/\*

SUBMITTED BY: HANIA IMTISAL

approach: count the list and maintain two pointers, make one go through kth iterations. When done, move both the pointers so that 1 pointer.next corresponds to new head and 1 pointer corresponds to the last element of prev list. then do the shifting by making the 2 pointer that corresponds to last join the 1 element of old list to make it circular and split the list by making the pointer poitning to the new head and making it's previous null.\*/

/\*\*

\* @param {ListNode} head

\* @param {number} k

\* @return {ListNode}

\*/

var rotateRight = function(head, k) {

if (!head || k === 0) return head;

let count = 0;

curr = head;

while (curr) //count the number of nodes.

{

count++;

curr = curr.next;

}

k = k%count; //checking boundry limits if k is within the list length

let prev = head;

curr = head;

while(k--)

{

curr = curr.next; // move this to node where split will occur

}

while(curr.next) // move this till it reaches the curr reaches end of list and prev points

{

prev = prev.next;

curr = curr.next;

}

curr.next = head; // make the list circular by pointing last node to start

head = prev.next; // prev.next corresponds to the new head as prev points to the last node of

//new list

prev.next = null;

return head;

};