Qn Link : <https://leetcode.com/problems/continuous-subarray-sum>

Observation :

* Remainders of the sum be 0 , 1 , 2 , 3 , ……… , k – 1
* Suppose a remainder of the subarray is “X” and after sometime again the subarray remainder is “X” then there exists a subarray with modulo K.
* If a subarray whose sum is modulo by K means , then the remainder will repeat

arr = [23, 2, 4, 6, 7] , k = 6

{23} % 6 = 5;

{23 + 2} % 6 = 1;

{23 + (2 + 4)} % 6 = 5;

* We again got the remainder 5 , so there is a sum which added to 23 is a modulo of 6 .

class Solution {

    public boolean checkSubarraySum(int[] nums, int k) {

        int n = nums.length;

        int sum = 0;

        Map<Integer , Integer> map = new HashMap<>();

        for(int i = 0 ; i < n; i++){

           sum += nums[i];

           int rem = sum % k;

           // If the remainder is zero and length is >= 1 (0 based indexing)

           if(rem == 0 && i >= 1)

            return true;

            //If remainder already exists , then the sum somwhere bt is modulo by K

           if(map.containsKey(rem)){

            int length = i - map.get(rem);

            //Even if the sub array sum exists , chack whether its length is >= 2

            if(length >= 2){

                return true;

            }

           }else

           //If the element is not present , put the element in the array

            map.put(rem , i);

        }

        return false;

    }

}