3 SUM PROBLEM

* In this problem, we are supposed to find the 3 elements in an array which add up to a given sum (without repetitions)

Brute force solution is to try out all triplets and check if their sum adds up to the required amount. We can store each answer array in a set to avoid duplicates.

Better solution involves the use of hashing. We create a hashmap or hasharray for the input array. Then after running 2 loops for i and j, we will try to look for remaing difference in the hashmap.

Optimal Solution involves the use of 3 pointers i, j & k. i will be at the start, j will be at i + 1 and k will be at the end. We will try a similar approach to the 2 sum problem where if our desired sum is greater than current, then we move j++ and if it is lesser, we move k--. We move both j++ and k-- once we find an answer and i is changed once j and k cross each other.

Something worth noting is that we have to sort the array at the start.

Pseudocode :

```
threeSumProblem (arr , N , K) {
    int ans []
    for(i = 0 → N) {
        if(i > 0 && arr[i] == arr[i-1]) {
            continue
        }
        j = i + 1 , k = N - 1
        while(j < k) {
            if (arr[j] + arr[k] + arr[i] < K) {
                j++
            } else if ( arr[i] + arr[j] + arr[k] > K){
                k--
            } else {
                ans.append({arr[i] , arr[j], arr[k]})
                boole = true
                while (boole) {
                    if(arr[j] == arr[j+1]) {
                        j++
                    } else {
                        boole = false
                    }
                }
                boole = true
                while (boole) {
                    if(arr[k] == arr[k-1]) {
                        k--
                    } else {
                        boole = false
```

```
        }
      }
    }
  }
  return ans
}
```