__/__/____

Hashing

① Find the frequency of an element in an array

Brute force

- you get an input array
- get total no of queries
- get each query i
- compare the query with the whole array
- So for every query check with each element of the whole array
- using a counter print out the frequency of the query

:- Takes too much time O(Q x N)

Semi optimized (Hash array)

- get an input array
- create a hash array (ideally a size of the largest elem in the ip array)

  ex: i/p - | 7 | 6 | 4 | 3 | 2 |

  hash array [13] = {0}
- and all elements in hash array initially assigned to 0
- Now map the elements of I/P array as indexes of hash array & store each elements count

  hash [arr [i]] ++

__/__/____

	0	1	2	3	4	5	6	7
arr:	7	2	7	3	1	4	5	5

hash[9]	0	1	2	3	4	5	6	7	8	9
	0	1	1	1	1	2	0	2	0	0

i/p = 7 -> hash[7] = 2
i/p = 5 -> hash[5] = 2
i/p = 3 -> hash[3] = 1

- Then just get the query
- & print hash[q]

:- This even tho takes a bit less time its not optimal if array has large number elements.

Optimized (Hashmap)

- get the i/p array
- create an unorderd map with key as the i/p array elements and the keys values to be the count of the elements
  mp[arr[i]]++ [here also initially values will be 0]
- Then get the queryes
- & print mp[q]

[Unorderd map cuz of the array eles order won't be sorted but in orderd it will be sorted]

② character frequency

Brute force

- get a string input
- get the number of queries
- keep a counter to count frequency
- get the queries
- compare the query character with each character of the string
- & print out count

optimized (unordered map)

- get a string input
- create an unordered map of char & int
  char as key
  int for frequency
  mp[s[i]]++
- get the queryies
- print mp[q]

③ largest & smallest frequency

- get the input array
- store the frequencies in the map
- now take 4 variables to keep track of the element & the frequency
  i.e - largestelement = 0 , largest freq = 0
  here both assigned to 0 as 0 is the least and any frequency and element will be greater than 0 then u update the two variables

- leastele = 0 , leastfreq = n

  ↙ Since you're not comparing element its fine with 0

  ↓ n coz any element in the ip array can max be of n times. So the frequencies is compared with this initially.

- now iterate through the map & check 2 conds

```
if (count < leastfreq)
{
    leastfreq = count; -> it.second
    leastele = element -> it.first
}
if (count > largestfreq)
{
    largestfreq = count;
    largestelem = element;
}
```

- Then print out both elements outside the iterator.