Date: / /

Hashing

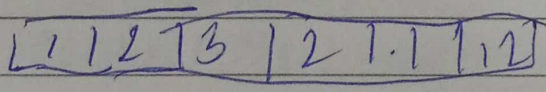
→ Pre storing & fetching

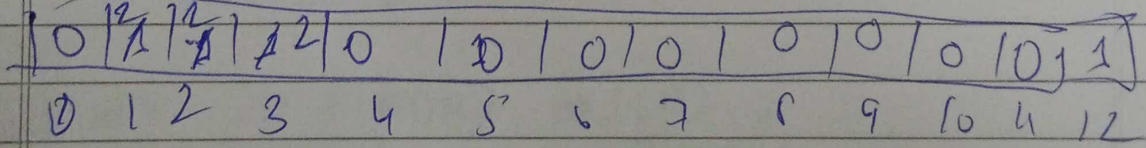
i.e. I will make an pre array or hash array with 0 filled.
Now I will tell how many nos I have 1 or 2 in my asked array.

Then ask how many 1's or 2's from array I have

Given = size of array
: array
= no. of queries
= query no.

hash[n] → n = max size that que^n will create like 13 or 10^5





Date __/__/____

```
int main (){
    int n;
    cin>>n;
    int arr[n];
    for(int i=0; i<n; i++){
        cin>>arr[i];
    }

    // pre compute

    int hash[13]={0};
    for(int i=0; i<n; i++){
        hash[arr[i]] += 1;
    }

    int q;
    cin>>q;
    while(q--){
        int number;
        cin>>number;
        cout<<hash[number]<<endl;
    }
    return 0;
}
```

- integer array

  inside main max array size declarable $< 10^6$

  but if ques limit is greater than this declare it globally till 10^7.

Date.__/__/____

for string

ASCII 'a' = 97
'z' = 122

for hashing

'f' = 102
'a' = 97
so hash value = 102 - 97 = 5

```
hash[256];
for(int i=0; i<s.size(); i++){
    hash[s[i]-'a']++;
}
```

```
while(q--){
    char c;
    cin>>c;
    cout<<hash[c-'a']
}
```

In case of lowercase
In case of uppercase [c-'A']

if lowercase is not mentioned

```
int hash[256]
for(int i=0; i<s.size(); i++){
    hash[s[i]]++
}
```

```
while(){
    cout<<hash[c]
}
```

In case matrix is mentioned

Date __/__/____

for tackling the int array > 10^7

we use maps / unordered-map

map = key : value
key → number, value → frequency

arr → 1, 2, 3, 1, 3, 2, 12

mpp[1]++
mpp[2]++
mpp[3]++
mpp[1]++
mpp[3]++
mpp[2]++
mpp[12]++

map
12 → 1
3 → x2
2 → x2
1 → x2

cout << mpp[4] << → 0 since there is no 4 in arr

→ Beneficial than hash maps as when to store elements we need to create hash[13] but here only no. of unique elements present are stored so no extra memory used

```
pre-compute
map<int, int> mpp;
for(int i = 0; i<n; i++){
    mpp[arr[i]]++;
}
while{
    cout << mpp[number];
```

Date: / /

for string

map<char, int> mpp;

mpp[s[i]]++

Time complex of ordered mapp
O(log n)

but in unordered_map
avg & best
O(1)
but in worst case (very rare)
O(N)

first use unordered. if it gives tle then use ordered