

**TCS NQT 2025 -26th march exact coding questions.(both shifts)**

Learning With Ram

Q1)Find lcm of two numbers

product of number = hcf\* lcm;

so lcm = a\* b/gcd;

**Code:-**

```
#include<bits/stdc++.h>
```

```
using namespace std;
```

```
int find_gcd(int n1,int n2)
```

```
{
```

```
    int gcd=1;
```

```
    for(int i=1;i<=min(n1,n2);i++)
```

```
    {
```

```
        if(n1%i==0 && n2%i==0)
```

```
        {
```

```
            gcd =i;
```

```
        }
```

```
    }
```

```
    return gcd;
```

```
}
```

```
int main()
```

```
{
```

```
    int n1,n2;
```

```
    cin>>n1>>n2;
```

```
    cout<<(n1*n2)/(find_gcd(n1,n2));
```

```
    return 0;
```

```
}
```

## Q2) Problem Statement: Instant Runoff Voting System

### Description:

There are 5 candidates (A, B, C, D, E) participating in an election. Each voter ranks all 5 candidates in order of preference.

The election follows the Instant Runoff Voting (IRV) system:

Each voter's first preference is initially counted as their vote.

If any candidate receives more than 50% of the votes, they are declared the winner.

If no candidate has more than 50% of the votes, the candidate with the fewest first-choice votes is eliminated.

The votes for the eliminated candidate are redistributed to the next preferred candidate still in the race.

This process repeats until a candidate secures more than 50% of the votes.

### Input Format:

The first line contains two integers N (number of voters) and M (number of candidates, always 5).

The next N lines contain a string of M uppercase letters representing the voter's ranked preferences.

### Output Format:

Print:

Winner <candidate\_name>

where <candidate\_name> is the candidate who wins the election.

### Constraints:

$1 \leq N \leq 1001$

M=5 (always 5 candidates)

Candidates' names are always uppercase letters: A, B, C, D, E.

Each voter provides a valid ranking containing all 5 candidates exactly once.

### Example Input 1:

```
45 5
ABCED
BCDAE
ACBED
CBDEA
```

Winner A

/\*

Read the input values: Number of voters (N) and candidate rankings for each voter.

Use a set to keep track of active candidates.

Count first-choice votes for active candidates.

Check if any candidate gets more than 50% of the votes.

If no candidate gets the majority:

Find the candidate with the least votes.

Eliminate them and redistribute their votes.

Repeat until a candidate gets a majority.

\*/

Code

```
#include <iostream>
```

```
#include <vector>
```

```
#include <unordered_map>
```

```
#include <set>
```

```
using namespace std;
```

```
// Function to determine the winner using Instant Runoff Voting (IRV)
```

```
string instantRunoffVoting(vector<string> &votes, int N) {
```

```
    set<char> activeCandidates = {'A', 'B', 'C', 'D', 'E'};
```

```
    while (true) {
```

```
        unordered_map<char, int> voteCount;
```

```
        // Initialize vote count for active candidates
```

```
for (char c : activeCandidates) {  
    voteCount[c] = 0;  
}  
  
// Count the first-choice votes  
for (const string &ballot : votes) {  
    for (char c : ballot) {  
        if (activeCandidates.count(c)) {  
            voteCount[c]++;  
            break; // Only count the first valid preference  
        }  
    }  
}  
  
// Check if any candidate has more than 50% of votes  
for (const auto &pair : voteCount) {  
    if (pair.second > N / 2) {  
        return string("Winner ") + pair.first;  
    }  
}  
  
// Find the candidate with the least votes  
char minCandidate = '\0';  
int minVotes = N + 1;  
  
for (const auto &pair : voteCount) {  
    if (pair.second < minVotes) {  
        minVotes = pair.second;  
        minCandidate = pair.first;  
    }  
}
```

```

        // Eliminate the candidate with the least votes
        activeCandidates.erase(minCandidate);
    }
}

int main() {
    int N, M;
    cin >> N >> M;

    vector<string> votes(N);
    for (int i = 0; i < N; i++) {
        cin >> votes[i];
    }

    cout << instantRunoffVoting(votes, N) << endl;
    return 0;
}

```

### Q3)Leftover elements

Problem Statement: Leftover Elements

Description:

You are given a range [L, U] and a list of N elements. The task is to find the leftover elements in the range [L, U] that are not present in the given list A. The output should group consecutive leftover elements into separate lists.

Input Format:

The first line contains three integers L, U, and N, where:

L: The lower bound of the range.

U: The upper bound of the range.

N: The number of elements in the list A.

The second line contains N space-separated integers representing the elements in A.

Output Format:

Print the leftover elements as separate groups of consecutive numbers enclosed in square brackets [ ].

Constraints:

$1 \leq L \leq U \leq 1000$

$0 \leq N \leq (U - L + 1)$

$L \leq A[i] \leq U$

example-1

1 10 1

5

1,2,3,4,5,6,7,8,9,10

output

[1 4] [6 10]

Explanation:

Given range: [1, 10]

List A: [5]

Leftover elements: [1, 2, 3, 4] and [6, 7, 8, 9, 10]

Group consecutive numbers: [1 4] and [6 10]

Example 2

Input:

3 90 5

7 22 50 66 78

output

[3 6] [8 21] [23 49] [51 65] [67 77] [79 90]

Explanation:

Given range: [3, 90]

List A: [7, 22, 50, 66, 78]

Leftover elements:

[3, 4, 5, 6]

[8 to 21]

[23 to 49]

[51 to 65]

[67 to 77]

[79 to 90]

These are printed as separate groups.

/\*approach

Read input values: L, U, N, and the list A.

Store given elements in a set for quick lookup.

Iterate through range [L, U]:

If an element is not present in A, start a new range.

If an element is present in A, close the previous range and store it.

Store the consecutive leftover numbers as ranges.

Print the ranges in the required format. \*/

**Code:-**

```
#include <iostream>
```

```
#include <vector>
```

```
#include <set>
```

```
using namespace std;
```

```
void findLeftoverElements(int L, int U, vector<int>& A) {
```

```
    set<int> present(A.begin(), A.end()); // Store given elements in a set
```

```
    vector<pair<int, int>> result;
```

```
    int start = -1; // Start of a range
```

```
    for (int i = L; i <= U; i++) {
```

```
        if (present.find(i) == present.end()) { // Not found in A
```

```
            if (start == -1) {
```

```
                start = i; // Start new range
```

```
            }
```

```
        } else {
```

```

        if (start != -1) {
            result.push_back({start, i - 1});
            start = -1;
        }
    }
}

if (start != -1) {
    result.push_back({start, U});
}

// Print the results
for (auto &range : result) {
    cout << "[" << range.first << " " << range.second << "]" ";
}
cout << endl;
}

```

```

int main() {
    int L, U, N;
    cin >> L >> U >> N;

    vector<int> A(N);
    for (int i = 0; i < N; i++) {
        cin >> A[i];
    }

    findLeftoverElements(L, U, A);
    return 0;
}

```

/\* Time Complexity:

$O(U - L + N) \rightarrow$  Efficient for large inputs. \*/

**Subscribe Learning With Ram youtube channel for supporting us**