Intra NSU Junior Programming Contest Summer 2023 - Editorial

Short analysis

Problem	Setter	Contributors	Total Solve	Difficulty	Solution
A - WHAT DID IT COST?	Tahmid Ahmed	Sarwar Alam Minhaj, Omar Haroon	62	Easy	Motins's reply was "NOTHING."
B - Domino Effect	Tahmid Ahmed	Muhammad Zubair, Farhan Omi	60	Easy	Count how many dominos will fall after the Kth domino has fallen, including the Kth domino.
C - Pagol Sort	Mahir Shahriar Tamim	Muhammad Zubair, Maharun Afroz, Farhan Omi, Tahmid Ahmed, Sahil Yasar	47	Easy	Sort the numbers in decreasing order.
D - Multiverse of Tahmid	Sahil Yasar	Fuwad Hasan, Muhammad Zubair	50	Easy-Medium	If the number is divisible by a and not divisible by b and c, then print "Tahmid Alam". If the number is divisible by b and not divisible by a and c, then print "Tahmid Ashraf". If the number is divisible by c and not divisible by a and b, then print "Tahmid Ahmed". If the number is divisible by any two or all three numbers, then print "Oh no".
E - World Cup	Muhammad Zubair	Sahil Yasar, Fuwad Hasan, Maharun Afroz	47	Easy-Medium	Print ($^{N}C_{2}$) + 3
F - Moye Moye	Muhammad Zubair	Tahmid Ahmed, Farhan Omi, Sahil Yasar, Omar Haroon, Fuwad Hasan	36	Easy-Medium	Count the number of substrings that contain "moye". If the count is zero, print "no fighting". If the count is even, print "moye moye". If the count is odd, print "moje more".
G - Flash Mob	Tahmid Ahmed	Farhan Omi, Sahil	3	Medium	Print "Yes" if no class time conflicts

		Yasar, Omar Haroon, Fuwad Hasan, Mahir Shahriar			with the flash mob time. Print “No” otherwise.
H - Beautiful NSU	Maharun Afroz	Nadman Ashraf Khan, Omar Haroon, Sahil Yasar	16	Medium	Find the combined area of the triangle ***ACE*** and ***BDF***. Subtract the overlapping area of these two triangles, if there is any.
I - Beautiful First Semester	Md Tanvirul Islam Niloy, Mahir Shahriar Tamim	Farhan Omi, Sahil Yasar	1	Medium	Store the messages with the priority of their respective courses, length of the message, and the input order of the message. Print the stored order of the queried message.
J - Long Division	Sarwar Alam Minhaj	Fuwad Hasan, Mahir Shahriar, Sahil Yasar	1	Medium-Hard	Find the prime factors up to 10^6. Store the count of prime factors of the given arrays. Compare the count of prime factors and print "No" if the count of any of the primes in the numerator is less than the count of that prime in the denominator. Otherwise, Print "Yes".
K - Aila Jaduu	BM Monjur Morshed, Omar Haroon	Mahir Shahriar, Tahmid Ahmed	0	Hard	Store the cumulative sum of the amount of bourn vita. Then, perform a binary search on the cumulative sum for each day.
L - Pengu Jor	Tahmid Ahmed	Fuwad Hasan, Mahir Shahriar, Sahil Yasar	3	Medium-Hard	Calculate the sum of all numbers from 1 to N that are not multiple of P or J without iterating through all the numbers.

Detailed analysis:

A - WHAT DID IT COST?

Problem setter - Tahmid Ahmed
Contributors - Sarwar Alam Minhaj, Omar Haroon
Total solves - 62
Difficulty - Easy
Approach -
This problem doesn't require analytical thinking. The problem statement asks to print a **single word,** and that is Motin's reply to Abbas when Abbas asked, "**WHAT DID IT COST?**". The question was not in the problem statement. Rather, it was in the attached meme where the answer to that question was "**NOTHING**." The solution is just to print "NOTHING" without taking any input.
To make it easier, "nothing" has been checked as a substring in the output without case sensitivity. So if your output has "nothing" as a substring, it was considered correct.

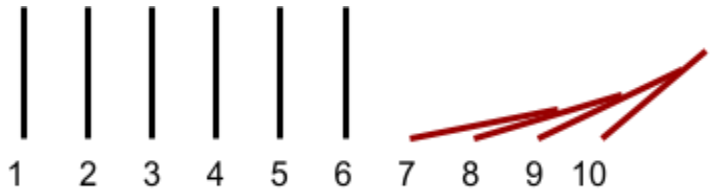
C++ Solution -

```
#include<bits/stdc++.h>
using namespace std;
int main(){
  cout << "NOTHING" << endl;
}
```

B - Domino Effect

Problem setter - Tahmid Ahmed
Contributors - Muhammad Zubair, Farhan Omi
Total solves - 60
Difficulty - Easy
Approach -

The solution to this printing is that the number of dominos will fall.
If n = 10 and k = 7, there are 10 dominos in a row, and the 7th domino has fallen, the 7th domino will fall and make the 8th fall. The 8th will make the 9th, and so on.



So the solution is to count how many dominoes are after the kth domino that is n - k; they will fall after k, and the kth will also fall, so we need to add 1 to the answer.
So the solution becomes n - k + 1

C++ Solution -

```
#include<bits/stdc++.h>
using namespace std;
int main(){
  int n, k;
  cin >> n >> k;
  cout << (n - k  + 1) << endl;
}
```

C - Pagol Sort

Problem setter - Mahir Shahriar Tamim
Contributors - Muhammad Zubair, Maharun Afroz, Farhan Omi, Tahmid Ahmed, Sahil Yasar
Total solves - 47
Difficulty - Easy
Approach -
The initial idea was to let any O(n^2) sorting algorithm to pass. (Bubble / Insertion Sort).
But we decreased the constraints to make the problem easier.
Hence, anyone unaware of any sorting algorithm can write the solution using if-else statements.

Solution **using If - Else Statement**
Solution **using Bubble Sort Algorithm**

D - Multiverse Of Tahmid

Problem setter - Sahil Yasar
Contributors - Fuwad Hasan, Muhammad Zubair
Total solves - 50
Difficulty - Easy-Medium
Approach -
You should become familiar with the % operator which is also called the modulo or remainder operator. And so if A%B is equal to 0, then B divides A, since there is no remainder.
With this out of the way, you only need 3 if statements to check for each Tahmid. So for Tahmid Alam, check if the number n is divisible by a and not divisible by b and c. Do the same for Tahmid Ashraf and Tahmid Ahmed, where the number will only be divisible by b and c, respectively. If none of the if statements are satisfied, then it is obvious that the number n is divisible by 2 or more of a, b, and c, since as written in the question, the number n is guaranteed that n is divisible by one of the numbers at least. And so print "Oh no".

Solution -
C Solution
C++ Solution

E - World Cup

Problem setter - Muhammad Zubair
Contributors - Sahil Yasar, Fuwad Hasan, Maharun Afroz
Total solves - 47
Difficulty - Easy-Medium

Approach - To calculate the number of matches such that each team competes with every other team exactly once, we can use $^{N}C_{2}$ = N * (N - 1) / 2. Also, there will be two semifinals and one final match. So add 3 to the answer.

In a scenario where each team competes against every other team only once, we need to determine the number of ways to choose two teams for a single match. Let's describe this with an example of five teams: **a, b, c, d,** and **e.** (Here, n = 5)

First, we want to find the number of unique pairings for matches:

- For team 'a,' we can pair it with each of the other teams: (a vs b), (a vs c), (a vs d), and (a vs e).
- For team 'b,' we pair it with the remaining teams it hasn't faced yet: (b vs c), (b vs d), and (b vs e).
- For team 'c,' we pair it with the remaining teams it hasn't faced: (c vs d) and (c vs e).
- For team 'd,' we pair it with the last remaining team it hasn't faced: (d vs e).

This results in a total of 10 unique pairings, and this can be calculated using the formula for combinations, ${}^{N}C_{2}$ where n is the number of teams: ${}^{N}C_{2}$ = N * (N - 1) / 2

Plugging in n = 5, we get: ${}^{5}C_{2}$ = 5 * (5 - 1) / 2 = 10

So, there are 10 unique pairings of teams for matches.
Next, we consider that there will be two semifinals and one final match. So, we add 3 matches to the 10 unique pairings: 10 (unique pairings) + 3 (2 semifinals and 1 final) = 13 matches in total.

Therefore, there will be a total of 13 matches, including the 10 unique pairings or round-robin group stage matches and the three knockout matches for the semifinals and the final.

C Solution -

```c
#include <stdio.h>

int main() {
  int t;
  scanf("%d", &t);
  while (t--) {
    int n;
    scanf("%d", &n);
    int ans = (n * (n - 1) / 2) + 3;
    printf("%d\n", ans);
  }
  return 0;
}
```

F - Moye Moye

Problem setter - Muhammad Zubair
Contributors - Tahmid Ahmed, Farhan Omi, Sahil Yasar, Omar Haroon, Fuwad Hasan
Total solves - 36
Difficulty - Easy-Medium

Approach - The problem is about counting the occurrences of the substring **moye** within a given string. To do this, iterate through the string from index 0 to n - 3 (exclusive) and increment a 'count' variable whenever you encounter a sequence where the character at position i is 'm', at position i + 1 is 'o', at position i + 2 is 'y' and at position i + 3 is 'e'.

- If the count of 'moye' substrings is equal to zero, you should output "no fighting."
- If the count is an even number, you should print "moye moye."
- If the count is an odd number, you should display "moje more."

C Solution -

```c
#include <stdio.h>

int main() {
  int t;
  scanf("%d", &t);
  while (t--) {
    int n;
    scanf("%d", &n);
    char s[n + 1];
    scanf("%s", s);
    int count = 0;
    for (int i = 0; i < n - 3; ++i) {
      if (s[i] == 'm' && s[i + 1] == 'o' && s[i + 2] == 'y' && s[i + 3] == 'e') {
        ++count;
      }
    }
    if (count == 0) printf("no fighting\n");          // if count is zero
    else if (count % 2 == 0) printf("moye moye\n");   // if count is even
    else printf("moje more\n");                       // if count is odd
  }
  return 0;
}
```

G - Flash Mob

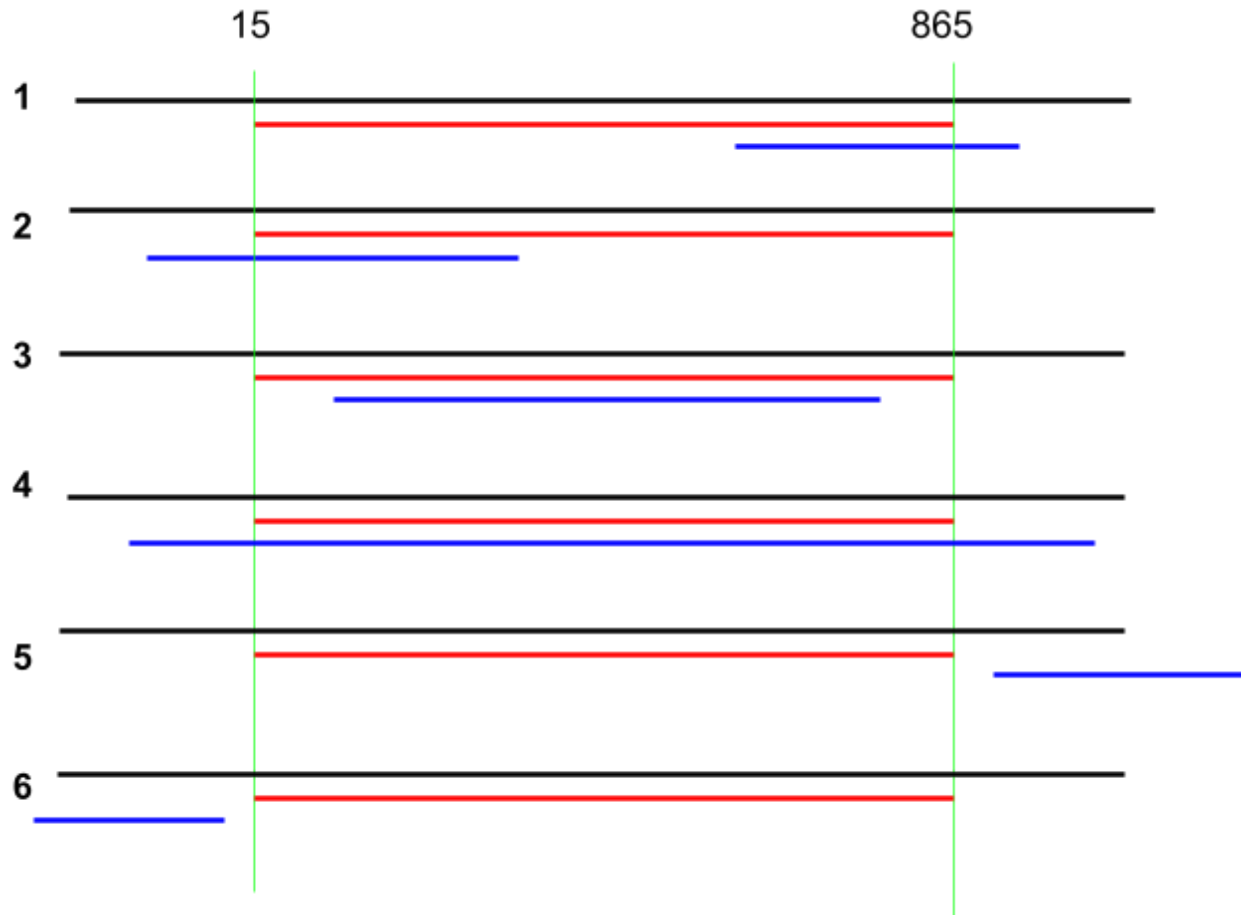
Problem setter - Tahmid Ahmed
Contributors - Farhan Omi, Sahil Yasar, Omar Haroon, Fuwad Hasan, Mahir Shahriar
Total solves - 3
Difficulty - Medium
Approach -
The solution is to check if any class clashes with the flash mob time. To do so, one easy way is to convert the time into minutes of that day. This works because it was guaranteed that the start and end times would be on the same day. To get what minute a time of the day, multiply the hour by 60 and add the remaining minutes.
For example,
if the time is 00:15, it is 0 * 60 + 15 = 15th minutes of that day.
If the time is 14:25, it is 14 * 60 + 25 = 840 + 25 = 865th minute of the day.

After converting the start time and end time of the flash mob, you get a window of time. Now, no class time should overlap with that window to enjoy the flash mob.

So if the flash mob time is 00:15-14:25, the flash mob window is from 15 to 865 minutes of that day.

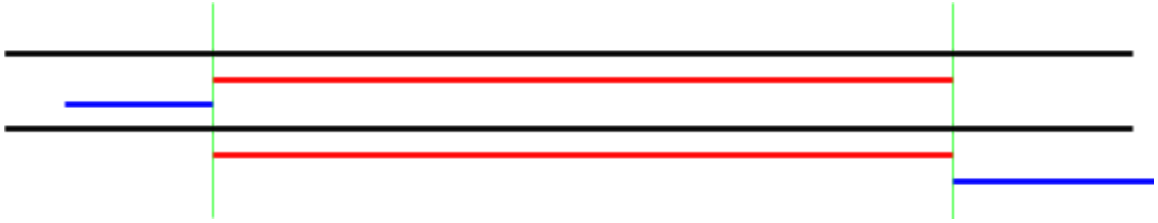
Let's consider
Black Line - Actual timeline
Redline - The flash mob is happening
Blue line - A class is happening.

The green line shows the time window of the start and end time of the flash mob.



The above 6 types of cases can happen in the given situation considering one class to check if it has a time conflict.

- In the first case, the class starts when flashmob is happening. **Conflict**
- In the second case, the class ends when the flash mob is happening. **Conflict**
- In the third case, the full class is between the flash mob. **Conflict**
- In the fourth case, the full flash mob is between the class. **Conflict.**
- In the fifth case, the full class is after the flash mob. **No conflict.**
- In the sixth case, the full class is before the flash mob. **No conflict.**



Note from the above image that if the blue line and red line both touch the same green line, it will also be a conflict. This means that at the same very minute, class is ending, and flash mob is starting, or flash mob is ending at the same minute class is starting.

Now you can put conditions to check if a class has a conflict with the flash mob like
If the class starts between the flash mob, or the class ends between the flash mob, or the whole flash mob is between the class, it is a conflict.

Another way of saying this is
If the flash mob starts between the class, or the flash mob ends between the class, or the whole class is between the flash mob. It is a conflict.

C++ Solution

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

using namespace std;

int inMinute(const string str) { // takes time in HH:MM and returns minute
  int hour = ((str[0] - '0') * 10) + (str[1] - '0');
  int minute = ((str[3] - '0') * 10) + (str[4] - '0');
  return (hour * 60) + minute;
};

int main() {
  init();
  int t;
  cin >> t;
  for (int tc = 1; tc <= t; ++tc) {

    int n;
    cin >> n;
    string flashmob_time;
    cin >> flashmob_time;

    int fs = inMinute(flashmob_time.substr(0, 5));
    int fe = inMinute(flashmob_time.substr(6, 5));

    bool conflict = false;

    for (int i = 0; i < n; ++i) {
      string class_time;
      cin >> class_time;
      int cs = inMinute(class_time.substr(0, 5));
```

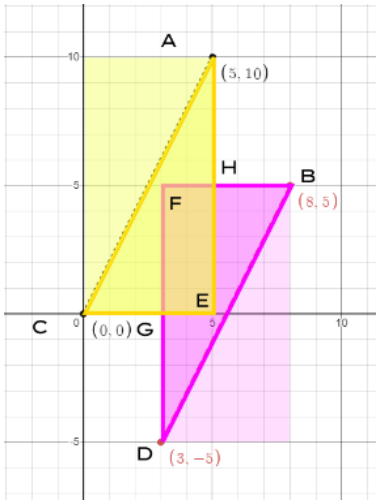
```
      int ce = inMinute(class_time.substr(6, 5));
      if ((cs <= fs and fs <= ce) or // starts between class
        (cs <= fe and fe <= ce) or // ends between class
        (fs <= cs and ce <= fe) // whole class between flashmob
      ) {

        conflict = true;
      }
    }
    cout << "Case " << tc << ": " << (conflict ? "No" : "Yes") << endl;
  }
}
```

Alternative Solution:
Case A: If the start and end time is **before** the flashmob time, then we can ignore it.
Case B: If the start and end time is **after** the flashmob time, then we can ignore it.
Otherwise, we have a conflict.
https://ideone.com/Vmlill (Written in C)

H - Beautiful NSU

Problem setter - Maharun Afroz
Contributors - Nadman Ashraf Khan, Omar Haroon, Sahil Yasar
Total solves - 16
Difficulty - Medium
Approach -
$\mathbf{ACE}$ and $\mathbf{BDF}$ are similar looking and have the same area(as mentioned in question).
So, they have the same base and height.



Refer to the image for better understanding.

```
base = (ax - cx)
```

```
height = (ay - cy)
```

```
ACE + BDF = 2 * (0.5 * base * height)

ACE + BDF = base * height
```

Now, as for the overlapping area,

```
if (ax > dx && by > cy)
{
    // Overlaps
    rec_base = (ax - dx)
    rec_height = (by - cy)

    Overlapping area = rec_base * rec_height
}
```

C++ Solution

tinyurl.com/injpc23-beautiful-nsu

or

tinyurl.com/injpc23-beautiful-nsu-diff-sol

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int t;
    cin >> t;

    for (int tc = 1; tc <= t; tc++) {
        /// Take input point A, C, B, D
        int ax, ay, cx, cy, bx, by, dx, dy;
        cin >> ax >> ay >> cx >> cy >> bx >> by >> dx >> dy;

        /// Calculate combined area of ACE and BDF
        int base = (ax - cx);
        int height = (ay - cy);

        int combinedArea = base * height;

        /// Calculating overlapping area
        int overlappingArea = 0;

        if (ax > dx && by > cy) // Overlaps
        {
            int rec_base = (ax - dx);
            int rec_height = (by - cy);

            overlappingArea = rec_base * rec_height;
        }

        /// Calculating answer
        // area = ACE + BDF - FHGE
        int area = combinedArea  - overlappingArea;

        cout << "Case " << tc << ": " << area << '\n';
    }
}

// To conquer CP
```

```
// https://tinyurl.com/cp-templates
```

I - Beautiful First Semester

Problem setter - Md Tanvirul Islam Niloy, Mahir Shahriar Tamim
Contributors - Farhan Omi, Sahil Yasar
Total solves - 1
Difficulty - Medium
Approach - While taking the input, set priority of the messages with respect to their course codes.
CSE115 being the highest (1) and BEN205 being the lowest (5).
The messages are then sorted using a **modified bubble sort**. The sorting algorithm prioritizes by:

Course priority: Lower numerical values indicate higher priority.
Message length: Shorter messages take precedence over longer ones if the priority is the same.
Original order: If priority and length are the same, the messages are left in the order they were received.
This sorting has a time complexity of O(n^2) due to the use of bubble sort, but N is 1000 so it will pass easily.

For each query, you can search linearly through the sorted list of messages to find a match and output the 1-based index of the message in the sorted list, representing the minute Patowary will reply to that message. This will take O(q * n).

Solution -
Solution in C
Solution in C++

Bonus question: Try solving for 1 <= N, Q <= 1e5

J - Long Division

Problem setter - Sarowar Alam Minhaj
Contributors - Fuwad Hasan, Mahir Shahriar, Sahil Yasar

Total solves - 1
Difficulty - Medium-Hard
Approach -
Any positive integer N can be expressed in terms of its prime factorization as follows:
$N = p_1^{e_1} \times p_2^{e_2} \times \ldots \times p_k^{e_k}$
Here, k represents the count of distinct prime numbers in the prime factorization of N. For instance: $12 = 2^2 \times 3^1$, $105 = 3^1 \times 5^1 \times 7^1$, and $68600 = 2^3 \times 5^2 \times 7^3$, and so on.

Consider two numbers in the form a^p and a^q. We can use the laws of indices to perform multiplication and division: $a^p \times a^q = a^{p+q}$ and $a^p \div a^q = a^{p-q}$.

For a^p to be divisible by a^q, the exponent p in the numerator must be greater than or equal to the exponent q in the denominator ($p \geq q$). This condition allows the part in the denominator to cancel out, leaving only a^{p-q} in the numerator. Since $p \geq q$, $p - q \geq 0$, hence a^{p-q} must be an integer. For example, 2^6 is divisible by 2^4, but 2^4 is not divisible by 2^5.

Using this idea, we can multiply numbers both in the numerator and the denominator. Then, for each prime in the denominator, we check if its exponent in the numerator is greater than or equal to that in the denominator. If this holds true for all prime numbers in the denominator, everything in the denominator will cancel out, leaving only prime numbers with non-negative exponents in the numerator.

To efficiently compute the prime factorization of numbers up to 10^6 in $O(\log n)$ time, we can use the Sieve of Eratosthenes to pre-calculate prime numbers.

Solution -
C++ Solution

K - Aila Jaduu

Problem setter - BM Monjur Morshed, Omar Haroon
Contributors - Mahir Shahriar, Tahmid Ahmed
Total solves - 0
Difficulty - Hard
Prerequisites to solve this problem: Prefix Sum and Binary Search
Approach -
Let's break down the problem step by step to understand the requirements and constraints:

- Rohit has 'n' packets of Bournvita with the 'i-th' packet containing 'ai' grams.

- He wants to drink 'bi' grams of Bournvita each day for 'm' days.
- If Rohit consumes all the Bournvita before 'm' days, Jaduu refills the packets while he sleeps.
- We need to calculate the number of packets left with Bournvita at the end of each day.

We can use a prefix sum array to store the cumulative Bournvita grams. This helps in determining the total Bournvita consumed up to any packet index 'i'.

For each day, we add the daily consumption 'bi' to a running total 'sum'. This 'sum' represents the total amount of Bournvita consumed up to that day.

We perform a binary search on the prefix sum array to find the index 'ind' such that 'prefix_sum[ind]' is just greater than 'sum'. This tells us the first packet that would be left with some Bournvita after consumption for the day.

The result 'res' is then calculated as the total number of packets 'n' minus 'ind', representing the packets left with Bournvita.

If 'ind' equals 'n', it means Rohit has consumed all packets, and therefore, all packets will be refilled. We reset 'sum' to 0, and 'res' is again 'n'.

The result 'res' is printed for each day.

Solution -
Author's Solution (C++)
Author's Solution (C)
Solution using Upper_Bound (C++) (uses the same concept)

L - Pengu Jor

Problem setter - Tahmid Ahmed
Contributors - Fuwad Hasan, Mahir Shahriar, Sahil Yasar
Total solves - 3
Difficulty - Medium-Hard
Approach -

The solution to this problem is to sum off all numbers between 1 and N except the numbers that are divisible by P. or J. To find the answer, the naive way would be to write a loop from 1 to N

like this.

```
sum = 0;
for(i = 1; i <= n; ++i){
    if(i % P != 0 and i % J != 0){
        sum += i;
    }
}
print(sum)
```

Look at the constraints of N. This loop will iterate for eternity. We need to optimize our solution.

We can calculate the sum of all numbers from 1 to N using a function of the sum of an arithmetic series. When you plug in a value for n, it calculates the sum of all the natural numbers from 1 to n, inclusive.

```
sum_fn(n){
    return (n* (n + 1)) / 2;
}
```

Lets take example case N= 15, p = 3, J = 5

```
{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 }
```

So, the total sum would be

```
total_sum = sum_fn(n)
```

Now, we must remove the sum of the numbers that are divisible by either P or J.

Step-1. The number of digits between 1 and N divisible by P is `N/P`, which are `1*P, 2*P ... (N/P)*P`.
And their sum is

```
=> 1*P + 2*P + ... (N/P)*P
=> P * (1 + 2 + ... (N/P))
=> P * sum_fn(N/P) = p_sum
```

```
total_sum = total_sum - p_sum
```

Step-2.The number of digits between 1 and N divisible by J is `N/J`, which are `1*J, 2*J ... (N/J)*J`.

And their sum is

```
=> 1*J + 2*J + ... (N/J)*J
=> J * (1 + 2 + ... (N/J))
=> J * sum_fn(N/J) =  j_sum
```

```
total_sum = total_sum - j_sum
```

In the example here, 15 is divisible by both 3 and 5, so if we subtract `j_sum + p_sum` from `total_sum`, 15 will be subtracted twice and make our answer less than the expected answer. If N is a large number, all numbers divisible by 3 and 5 will be counted twice. The sum of all numbers divisible by P and J has been subtracted twice. We need to add it back to the `total_sum`.

Notice that twice-counted numbers are common multiples of P and J, the lowest number is the Least Common Multiple (LCM) of P and J, and all above is a multiple of LCM(P, J).

Let denote

```
X = LCM(P, J)
```

Step-3. The digits between 1 and N divisible by X are `N/X`, which are `1*X, 2*X ... (N/X)*X`.
And their sum is

```
=> 1*X + 2*X + ... (N/X)*X
=> X * (1 + 2 + ... (N/X))
=> X * sum_fn(N/X) =  x_sum
```

Now, we add this to our total sum, which is our answer to the problem.

```
total_sum = total_sum + x_sum
```

C++ Solution

```cpp
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;

ll sum_fn(ll n) {
  return (n * (n + 1)) / 2;
```

```
}
ll lcm(ll a, ll b) {
  return (a / __gcd(a, b)) * b;
}
int main() {

  ll n, p, j;
  cin >> n >> p >> j;

  //Take the total sum
  ll total_sum = sum_fn(n);

  // Subtract sum of multiples of P
  total_sum = total_sum - (p * sum_fn(n / p));

  //Subtract sum of multiples of J
  total_sum = total_sum - (j * sum_fn(n / j));

  //Add sum of common multiples of P and J
  ll x = lcm(p, j);
  total_sum = total_sum + (x * sum_fn(n / x));

  cout << total_sum << endl;
}
```