# 2025 T2 Launch Week Chicken Contest Problems

UNSW CPMSoc

2 June 2025

## Contents

## Sum 90 [Maths]

Suppose two real numbers have a sum of 90. What is the maximum possible value of their product?

Enter your answer into the box below to submit and check your answer!

For all the tasks, including this one, you can submit as many times as you like with no penalty! Only your best submission will be counted.

## Joining Points [Maths]

Consider a $6 \times 6$ grid where points are located at integer coordinates $(x, y)$ where $0 \leq x, y \leq 5$.

If you select any two distinct points from this grid, what is the total number of different possible distances between them?

For all the tasks, including this one, you can submit as many times as you like with no penalty! Only your best submission will be counted.

## Wide Ruler [Maths]

An A4 sheet of paper measures 210 mm x 297 mm. What is the widest a 30 cm ruler can be which still fitting entirely on the page? Round your answer to the nearest mm.

For all the tasks, including this one, you can submit as many times as you like with no penalty! Only your best submission will be counted.

# Consecutive Counting [Maths]

A positive integer is called "egg-like" if it is the sum of 2 or more consecutive positive integers. For example, 22 is egg-like since $22 = 4 + 5 + 6 + 7$.

**Subtask 1 (40% of points)**

How many positive integers less than $10^6$ are not egg-like?

**Subtask 2 (30% of points)**

How many positive integers less than $10^{11}$ are not egg-like?

**Subtask 3 (30% of points)**

How many positive integers less than $10^{16}$ are not egg-like?

**Submission**

Submit your answer to the subtasks as a comma-separated list of integers. For example, if your answers to the subtasks are 1, 3 and 4, you should submit `1,3,4`. Note that if you have not solved a subtask, you can submit a dummy answer for that subtask. For example, if your answer to the first subtask is 1, you could submit `1,0,0`.

## Massive Mall [Maths]

After a long day's work, Jerry decided to reward himself with some food. As he entered a mall, he was struck with awe from it's size, and of how many restaurants were serving fried chicken.

The building has 13 elevators which each stop on at most 4 floors. If Jerry can go from any floor to any other floor using only one elevator, what is the maximum number of floors in the building?

For all the tasks, including this one, you can submit as many times as you like with no penalty! Only your best submission will be counted.

# Algebra 1 [Maths]

Find all functions $f : \mathbb{R} \to \mathbb{R}$ such that:

$$f(af(-b) + f(a - b)) + b = (b - 1)f(a)$$

for all real numbers $a, b \in \mathbb{R}$.

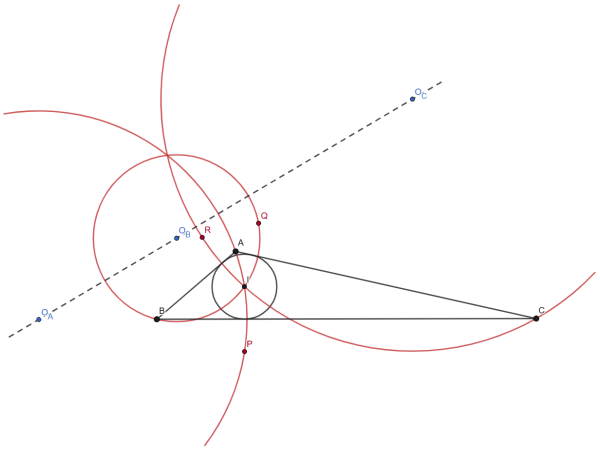
For all the tasks, including this one, you can submit as many times as you like with no penalty! Only your best submission will be counted.

# Geometry 1 [Maths]

Let $ABC$ be a scalene triangle with incentre $I$. Denote $P$, $Q$, and $R$ as the reflections of $I$ over lines $BC$, $AC$, and $AB$ respectively. Prove the centre of the circles $(PIA)$, $(QIB)$, and $(RIC)$ are colinear.

(Note: $(XYZ)$ is the circle passing through the 3 points $X$, $Y$, and $Z$).

For all the tasks, including this one, you can submit as many times as you like with no penalty! Only your best submission will be counted.

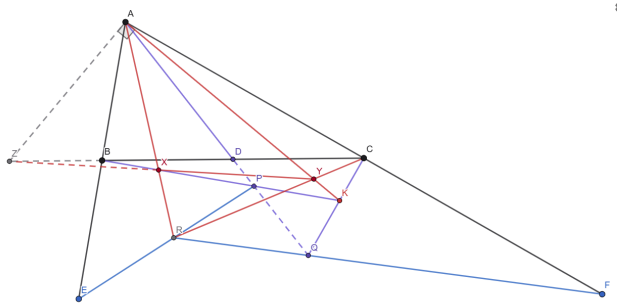


# Wild Chicken Chase [Maths]

For the past 2 hours, Frank has been chasing a chicken that snatched his wallet from him. Despite his weary state, he managed to corner it and retrieve his goods. However, he realized that along the way all of his coins fell out, sketching the path that they had been running around in...

Let ABC be a scalene triangle, $D$ be the midpoint of side $BC$ and $E, F$, the reflections of $A$ over points $B, C$. Let $K$ be the point such that $\angle ABK = \angle ACK = 90°$. Lines $BK$ and $CK$ intersect line $AD$ at $P$ and $Q$ respectively, and lines $EP$ and $FQ$ intersect at $R$. Let $AR$ and $KB$ meet at $X$, $AK$ and $CR$ meet at $Y$, and lines $BC$ $XY$ intersect at $Z$. Show that $\angle ZAK = 90°$.

For all the tasks, including this one, you can submit as many times as you like with no penalty! Only your best submission will be counted.



# Fair Digsum [Programming]

**Program time limit: 1 second**

**Program memory limit: 512 MB**

Alien chickens have just landed in Australia, and are struggling to learn the local slang. The first time they hear someone say "fair dinkum," the chickens mishear this as their alien mathematical concept, which they repeat back as "bok bawk," but when translated, becomes "fair digsum."

A positive integer $x$ is a fair digsum for a given $N$ if and only if $x = x\%N \times S(x) \times S(x)$, where:

- $x\%N$ represents the remainder of $x$ when divided by $N$, and
- $S(x)$ represents the sum of the digits in $x$.

Refer to the samples for examples of fair digsums.

The alien chickens threaten that they'll take all the chickens in the world to outer space, unless you can answer their question: For a given $N$, what are all the fair digsums less than or equal to an upper limit $M$?

### Input

The first and only line of input contains two integers, $N$, the number you will be dividing $x$ by, and $M$, the upper limit for your fair digsums.

You should read from standard input.

In Python, you could use the line `N, M = map(int, input().split())`.

In C or C++, you could use the line `int N, M; scanf("%d%d", &N, &M);`.

### Constraints

For all test cases:

- $2 \leq N \leq 200$,
- $1 \leq M \leq 1,000,000$.

### Output

You will need to output your answer onto multiple lines.

On the first line, output a single integer $K$, representing the total number of fair digsums. If there are no fair digsums, output `0`.

Over the following $K$ lines, output one digsum per line. Outputs must be given in increasing order.

You should write to standard output.

In Python, you could use the following code:

```
print(len(fairDigsums))
for x in fairDigsums:
    print(x)
```

In C or C++, you could use the following code:

```
printf("%d\n", &fairDigsums.size());
int i;
for (i = 0; i < fairDigsums.size(); i++) {
  printf("%d\n", &fairDigsums[i])
}
```

### Sample Input

```
100 2345
```

### Sample Output

```
8
1
10
405
605
810
1215
1620
2025
```

### Explanation

There are 8 fair digsums for $N = 100$ less than or equal to 2345. Some of these are explained as follows:

- For $x = 1$, the remainder when dividing by 100 is 1, and the digit sum is just 1. Therefore, it is a fair digsum since $1 = 1 \times 1 \times 1$.
- For $x = 405$, the remainder when dividing by 100 is 5, and the digit sum is $4 + 0 + 5 = 9$. Therefore, it is a fair digsum since $405 = 5 \times 9 \times 9$.
- For $x = 2025$, the remainder when dividing by 100 is 25, and the digit sum is $2 + 0 + 2 + 5 = 9$. Therefore, it is a fair digsum since $2025 = 25 \times 9 \times 9$.
- The next fair digsum for $N = 100$ would be $x = 2430$, but this is greater than 2345, so is not included.

### Scoring

Your program will be run on the sample case and 9 secret cases one after another, and if it produces the correct output for **all** test cases, it solves this task. Recall that your final score on the task is the score of your highest scoring submission.

# Stupid Dot Product [Programming]

**Program time limit: 1 second**

**Program memory limit: 512 MB**

Charlie the chicken is given two strips of paper. Each strip of paper is an array of $N$ integers. However, Charlie dislikes when the dot product of the arrays isn't zero, so he wants to peck at numbers on the paper. Each time Charlie pecks a number, it gets replaced with another integer of his choosing.

Given the arrays $A$ and $B$, what is the least number of pecks Charlie needs to make the dot product 0?

Note that numbers in the arrays can be negative, zero, or positive. The constraint only applies to initial values, meaning that the updated values can be outside of the range.

### Input

- The first line of input contains an integer $N$, representing the number of integers in each array.
- The second line contains $N$ space separated integers, representing $A$.
- The third line contains $N$ space separated integers, representing $B$.

You should read from standard input.

In Python, you could use the following code.

```
N = int(input())
A = list(map(int, input().split()))
B = list(map(int, input().split()))
```

In C or C++, you could use the following code.

```
int N; scanf("%d", &N);
int A[N]; for (int i = 0; i < N; i++) scanf("%d", &A[i]);
int B[N]; for (int i = 0; i < N; i++) scanf("%d", &B[i]);
```

### Constraints

For all test cases:

- $1 \leq N \leq 200,000$,
- $-100 \leq A_i, B_i \leq 100$ for $1 \leq i \leq N$.

### Output

Output a single integer, which represents the least number of pecks Charlie needs to make.

You should write to standard output.

In Python, you could use the line `print(answer)`.

In C or C++, you could use the line `printf("%d\n", answer);`.

### Sample Input 1

```
6
1 2 3 5 5 6
1 1 1 0 5 4
```

### Sample Output 1

```
1
```

### Explanation 1

The dot product can be made into 0 with just 1 change. One way to do this is to change $B_4$ from 0 into $-11$, which will make the updated dot product 0.

### Sample Input 2

```
2
-7 15
13 -43
```

### Sample Output 2

```
2
```

### Explanation 2

One way to make the dot product into 0 is to change $A_1$ from $-7$ to 0 and $A_2$ from 15 to 0.

### Scoring

Your program will be run on both sample cases and 13 secret cases one after another, and if it produces the correct output for **all** test cases, it solves this task. Recall that your final score on the task is the score of your highest scoring submission.

# Frog Pond [Programming]

**Program time limit: 1 second**

**Program memory limit: 512 MB**

You are wandering through the well known Sydney Park, where in order to reach one area to the other, you need to use a bridge. Sydney Park consists of $N$ areas besides frog pond and every bridge has a certain weight limit. You want to find the best path to the mysterious **Frog Pond**, which is always located at area 0. However, you're starting your journey from different locations throughout the park while carrying a very heavy backpack (To enjoy the scenery of Sydney park, he is carrying his portable tent and other camping tools).

Each bridge connects two areas and has a maximum weight it can support. For each of your $Q$ starting points, determine the **maximum weight** that can be carried from that starting area to the Frog Pond (area 0) using the bridges, **such that all bridges on the path can support that weight**.

### Input

- The first line contains three integers $N$, $M$ and $Q$ — the number of areas (1 to $N$), the number of bridges and the number of queries.
- The next $M$ lines each contain three integers $u$, $v$, and $w$ — representing a bridge between areas $u$ and $v$ with a maximum weight limit of $w$.
- The next $Q$ lines each contain a single integer $A_i$ — the starting area ($A_i \neq 0$) for the $i$-th query.

You should read from standard input.

### Constraints

- $2 \leq N \leq 10^5$
- $N \leq M \leq 5 \times 10^5$
- $1 \leq Q \leq N$
- $0 \leq u, v, A_i < N$
- $1 \leq w \leq 10^9$

It is guaranteed that the graph is connected.

### Output

For each query, print a single integer: the maximum weight that can be carried from area $A_i$ to the Frog Pond (area 0) using the available bridges. If there is no path from $A_i$ to 0, print `0`.

You should write to standard output.

### Sample Input

```
5 5 5
1 0 30
2 0 40
3 1 10
4 2 20
5 4 50
1
2
3
4
5
```

### Sample Output

```
30
40
10
20
20
```

### Scoring

Your program will be run on the sample case and 13 secret cases one after another, and if it produces the correct output for **all** test cases, it solves this task. Recall that your final score on the task is the score of your highest scoring submission.

# Chicken Merger [Programming]

**Program time limit: 3 seconds**

**Program memory limit: 512 MB**

You are a chicken trainer with a unique ability to merge chickens! When two chickens of the same level are merged, they become a new chicken with a level one higher than their original level. This process can be repeated indefinitely, allowing you to create increasingly powerful chickens.

You start with $N$ chickens, each with an integer level. You must support $Q$ operations of the following types:

1. Add a chicken with level $A$
2. Remove a chicken with level $A$ (it is guaranteed that such a chicken exists)
3. Output the highest level chicken you can create by merging chickens currently in your collection

For operations of type 3, you only need to calculate and output the highest possible chicken level that could be achieved through merging - you don't need to actually perform the merges or modify your chicken collection.

### Input

The first line of input contains two integers $N$ and $Q$, where $N$ is the initial number of chickens and $Q$ is the number of operations.

The second line contains $N$ integers, representing the levels of the initial chickens.

Then follow $Q$ lines, each describing an operation:

- `1 A`: Add a chicken with level $A$
- `2 A`: Remove a chicken with level $A$
- `3`: Output the highest possible level achievable through merging

You should read from standard input.

### Output

For each operation of type 3, output a single integer representing the highest possible level that can be achieved by merging chickens currently in your collection.

You should write to standard output.

### Constraints

For all test cases:

- $1 \leq N, Q \leq 1\,000\,000$
- $1 \leq A \leq 10^9$ for all chicken levels
- It is guaranteed that when removing a chicken, at least one chicken of that level exists

Additionally:

- For Subtask 1 (50% of points): $N, Q \leq 1\,000$
- For Subtask 2 (50% of points): there are no additional constraints

### Sample Input

```
3 5
1 1 2
3
2 1
3
1 2
```

```
3
```

### Sample Output

```
3
2
3
```

### Explanation

Initial state: [1, 1, 2]

1. Query highest possible level: You can first merge two level 1 chickens to get a level 2: $[2, 2]$, then merge two level 2 chickens to get level 3: [3]. You can't make any more merges, so the highest possible level is 3.
2. Remove a chicken with level 1: $[1, 2]$.
3. Query highest possible level: You can't make any merges, so the highest possible level is 2.
4. Add chicken with level 2: $[1, 2, 2]$.
5. Query highest possible level: You can first merge two level 2 chickens to get a level 3: $[1, 3]$. You can't make any more merges, so the highest possible level is 3.

### Scoring

For each subtask (worth 50% and 50% of points, as per the Constraints section), your program will be run on multiple secret test cases one after another, and if it produces the correct output for **all** test cases, it solves that subtask. Your program will receive the points for each subtask it solves. Recall that your final score on the task is the score of your highest scoring submission.

# Chicken Run [Programming]

**Program time limit: 1 second**

**Program memory limit: 512 MB**

You are a chicken running around a graph with $N$ vertices and $M$ directed edges. Each edge has a weight, which may be positive, negative, or zero.

You start at vertex 1 with score 0. When you move along an edge, its weight is added to your score, however your score cannot go below 0. More precisely, if your score is $S$, and you move along an edge of weight $w$, then your score becomes $\max(0, S + w)$.

What is the maximum score you can achieve after making exactly $K$ moves?

### Input

- The first line of input contains three integers $N$, $M$ and $K$, where $N$ is the number of vertices, $M$ is the number of edges, and $K$ is the number of moves.
- Then follow $M$ lines, the $i$th of which contains three integers $u_i$, $v_i$ and $w_i$, indicating that there is a directed edge from vertex $u_i$ to vertex $v_i$ with weight $w_i$.

You should read from standard input.

In Python, you could use the following code.

```
N, M, K = map(int, input().split())
edges = [tuple(map(int, input().split())) for _ in range(M)]
```

In C or C++, you could use the following code.

```
int N, M, K;
scanf("%d%d%d", &N, &M, &K);
int u[M], v[M], w[M];
for (int i = 0; i < M; i++) {
    scanf("%d%d%d", &u[i], v[i], w[i]);
}
```

### Constraints

For all test cases:

- $2 \leq N \leq 100$.
- $1 \leq M \leq N(N-1)$.
- $1 \leq K \leq 10^9$.
- $1 \leq u_i, v_i \leq N$ for all $i$.
- $-10^9 \leq w_i \leq 10^9$ for all $i$.
- $u_i \neq v_i$ for all $i$. That is, there are no loops.
- $(u_i, v_i) \neq (u_j, v_j)$ for all $i \neq j$. That is, there are no multiple edges.
- There is at least one way to make $K$ moves starting from vertex 1.

Additionally:

- For Subtask 1 (30% of points): $K \leq 1000$.
- For Subtask 2 (20% of points): $w_i \geq 0$ for all $i$.
- For Subtask 3 (50% of points): there are no additional constraints.

### Output

Output the maximum score you can achieve.

You should write to standard output.

In Python, you could use the line `print(answer)`.

In C or C++, you could use the line `printf("%lld\n", answer);`.

### Sample Input

```
4 5 6
1 2 -3
2 1 6
1 3 -1
3 2 2
1 4 3
```

### Sample Output

```
16
```

### Explanation

An optimal sequence of $K = 6$ moves is as follows, starting from vertex 1 with score 0.

1. Move to vertex 2. Your new score is $\max(0, 0 - 3) = 0$.
2. Move to vertex 1. Your new score is $\max(0, 0 + 6) = 6$.
3. Move to vertex 3. Your new score is $\max(0, 6 - 1) = 5$.
4. Move to vertex 2. Your new score is $\max(0, 5 + 2) = 7$.
5. Move to vertex 1. Your new score is $\max(0, 7 + 6) = 13$.
6. Move to vertex 4. Your new score is $\max(0, 13 + 3) = 16$.

### Scoring

For each subtask (worth 30%, 20% and 50% of points, as per the Constraints section), your program will be run on multiple secret test cases one after another, and if it produces the correct output for **all** test cases, it solves that subtask. Your program will receive the points for each subtask it solves. Recall that your final score on the task is the score of your highest scoring submission.

# Tree Building [Programming]

**Program time limit: 5 seconds**

**Program memory limit: 512 MB**

You are given a rooted tree with $N$ nodes numbered from 1 to $N$. The tree has $N-1$ edges, and node 1 is the root. Each node $i$ has a value $X[i]$.

In one operation, you may select any two nodes $i$ and $j$. This operation costs $|X[i] - X[j]|$ units of energy and activates every edge along the path connecting nodes $i$ and $j$. You can perform this operation as many times as you wish.

Your task is to find, for each non-root node, the minimum amount of energy needed to activate all edges on the path from that node to the root.

### Input

The first line contains a single integer $N$ ($1 \leq N \leq 10^6$), the number of nodes in the tree.

The second line contains $N$ space-separated integers $X[1], X[2], \ldots, X[N]$ ($1 \leq X[i] \leq 10^9$), the values of the nodes.

The next $N-1$ lines describe the edges of the tree. Each line contains two space-separated integers $u$ and $v$ ($1 \leq u, v \leq N$), indicating an edge between nodes $u$ and $v$.

### Output

Output $N-1$ lines. The $i$-th line should contain a single integer: the minimum energy needed to activate all edges on the path from node $i+1$ to the root.

### Constraints

- $1 \leq N \leq 10^6$
- $1 \leq X[i] \leq 10^9$
- The input forms a valid tree with node 1 as the root

### Sample Input

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

### Sample Output

```
1
1
1
2
```

### Explanation

For node 2: We can activate the edge $(1, 2)$ by choosing nodes 1 and 2, spending $|6 - 7| = 1$ unit of energy.

For node 3: We can activate the edge $(2, 3)$ by choosing nodes 2 and 3, spending $|7 - 8| = 1$ units of energy.

For node 4: We can activate the edge $(3, 4)$ by choosing nodes 3 and 4, spending $|8 - 9| = 1$ units of energy.

For node 5: We can activate the edge $(4, 5)$ by choosing nodes 4 and 5, spending $|9 - 10| = 1$ unit of energy, and then activate the edge $(2, 3)$ by choosing nodes 2 and 4, spending $|7 - 8| = 1$ units of energy. The total energy spent is 2 units.