Tsuen Wan Government Secondary School

Olympiad in Informatics Team

2025/26 Team Selection Test

Editorial

Tsui Siu Fong

2 July 2026

Main Character

Me (NGGYU)

All the “I” / “me” / “my” in the problems and the editorial are all the same me!

Quick Note

Please refrain from copying any code in this editorial.

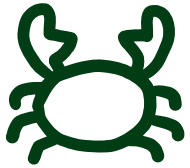
We strongly believe that you should solve the problems by your own.

You cannot improve from copying others' code.

With that said, let's begin!

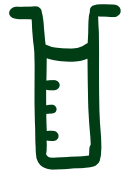
Overview

TS2601
Instant Summon



TS2602
Crab Me Crab

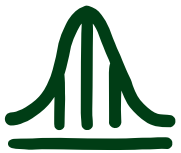
TS2603
Water Bird



TS2604
pH Value

TS2605
Zodiac Animals

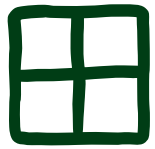
TS261
Five O Five



TS262
ConteStats

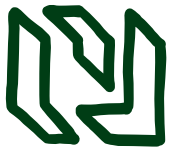
TS263
Work Past Password

TS264
CapitaliStatement



TS265
Window Installer

TS266
Love Leafer



TS267
Judge Addict

TS268
Mark Six

TS269
Candy World

Statistics

	Problem	Attempts	Full	Max	Mean	Std Dev	LQ	Median	UQ	Subtasks	First Solve	
TS2601	Instant Summon 即時新聞	32	10	10	9.438	1.952	10	10	10	2 | 31 · 4 | 31 · 4 | 29	Wang Matthew	00:01:13
TS2602	Crab Me Crab 蟹人聽聞	28	15	15	14.036	2.97	15	15	15	4 | 27 · 3 | 27 · 4 | 27 · 4 | 24	Wang Matthew	00:02:18
TS2603	Water Bird 一走鳥之	28	15	15	13	4.943	15	15	15	4 | 25 · 3 | 24 · 4 | 24 · 4 | 24	Wang Matthew	00:03:20
TS2604	pH Value 中和酸鹼	31	20	20	18.581	3.783	20	20	20	2 | 31 · 5 | 29 · 6 | 30 · 7 | 27	Fung Hiu Nok	00:02:48
TS2605	Zodiac Animals 十二生肖	29	40	40	31.793	15.189	33	40	40	8 | 25 · 18 | 23 · 14 | 22	Fung Hiu Nok	00:04:37
TS261	Five O Five 五五歸家	29	50	50	33.966	18.816	12	45	50	5 | 26 · 7 | 25 · 13 | 19 · 12 | 18 · 8 | 19 · 5 | 13	Wang Matthew	00:14:48
TS262	ConteStats 數據慢遊	23	60	60	45.391	22.577	33	60	60	6 | 14 · 3 | 4 · 6 | 17 · 3 | 2 · 36 | 17 · 18 | 2 · 12 | 14 · 6 | 4	Wang Matthew	00:18:28
TS263	Work Past Password 密密碼碼	20	65	65	44.65	28.606	7.5	65	65	5 | 14 · 13 | 14 · 11 | 15 · 4 | 15 · 15 | 13 · 17 | 13	Wang Matthew	00:20:44
TS264	CapitaliStatement 小題大作	18	75	75	50.556	27.753	38	60	75	8 | 15 · 19 | 15 · 11 | 15 · 26 | 9 · 7 | 10 · 4 | 9	Wang Matthew	00:26:06
TS265	Window Installer 鏗窗有力	19	75	75	57.474	29.671	19	75	75	4 | 17 · 6 | 16 · 9 | 16 · 9 | 14 · 47 | 14	Fok Kin Wing	00:29:32
TS266	Love Leafer 敬葉樂葉	16	75	75	36	34.916	0	17	75	4 | 7 · 6 | 7 · 15 | 7 · 11 | 7 · 14 | 7 · 17 | 10 · 8 | 7	Fok Kin Wing	00:25:33
TS267	Judge Addict 評測癮者	16	100	100	60.188	46.856	0	100	100	8 | 10 · 12 | 9 · 13 | 10 · 23 | 10 · 19 | 10 · 25 | 9	Fok Kin Wing	00:08:13
TS268	Mark Six 六合獎券	14	150	150	100.857	67.577	0	150	150	12 | 10 · 10 | 9 · 24 | 9 · 21 | 9 · 27 | 10 · 23 | 10 · 18 | 9 · 15 | 9	Fok Kin Wing	00:15:59
TS269	Candy World 糖果世界	12	250	250	107	105.23	17.5	56	250	5 | 10 · 10 | 8 · 15 | 8 · 18 | 4 · 13 | 6 · 32 | 7 · 32 | 5 · 32 | 4 · 65 | 4 · 13 | 4 · 15 | 4	Fok Kin Wing	01:22:40
		32	**1000**	**1000**	**366.125**	**334.225**	**87**	**267**	**618**			

Instant Summon

TS2601

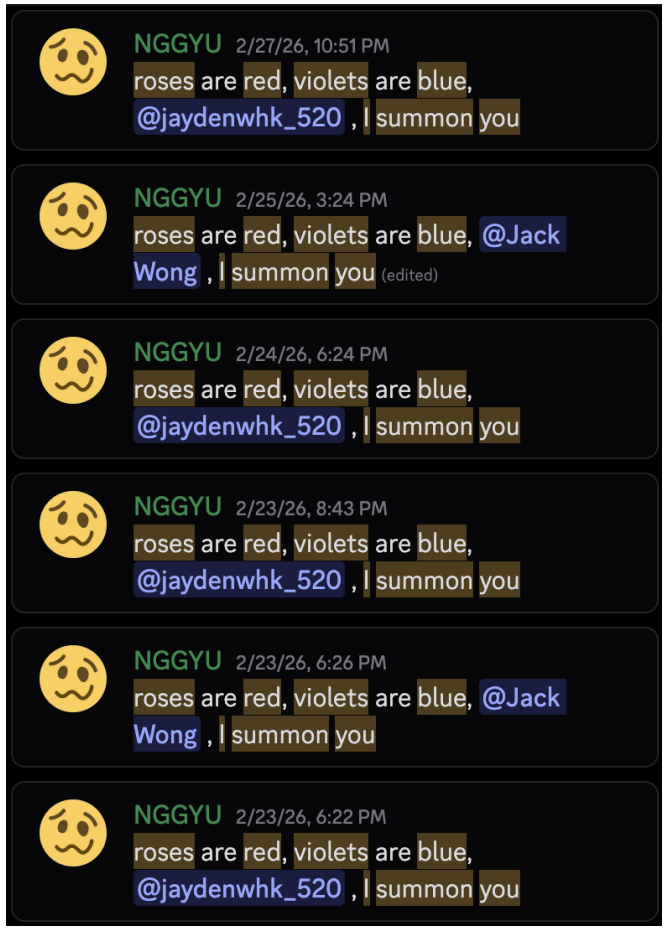
Problem Statement

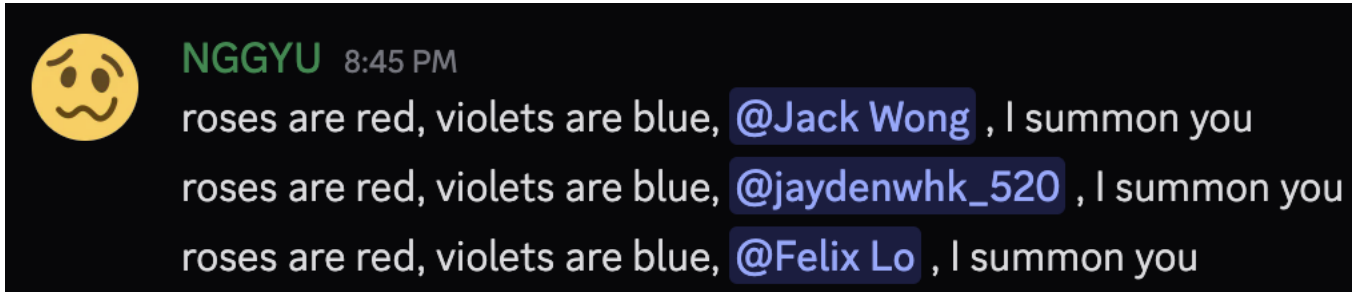
Given a string S, output

- `red` if S is `rose`
- `blue` if S is `violet`
- `summon you` if it S neither `red` nor `blue`

Fun Fact

The “magic spell” is actually used! (although it is not very effective…)





Subtask 1

$S =$ `rose`

Just output `red`.

How to output?

Use `cout` with double quotes for string type.

Subtask 2

$S =$ `rose` or `violet`

Output `red` if $S =$ `rose`, else `blue` (or `blue` if $S =$ `violet`, else `red`).

How?

Use if-else statement!

Full Solution

Now, S can be anything, not just `rose` or `violet`!

How do deal with it?

Use if-else statement with `if`, `else if` and `else`!

Simple!

Takeaway

In this problem, you need to output `red`, `blue` or `summon you`.

However, we noticed that some people typed it incorrectly and received a verdict of wrong answer. Luckily, they noticed their mistake, and they could successfully solve the problem.

To prevent this from happening, please copy important strings like these from the problem statement instead of typing them by yourself. It can save you a lot of headaches.

Common Mistakes

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

int main(){
    string s;
    cin >> s;
    if (s == "rose"){
        cout << "red";
    }else if (s == "violet"){
        cout << "blue";
    }else if (s == "hoki"){
        cout << "summon you";
    }else if (s == "jack"){
        cout<< "summon you";
    }
}
```

Note: The code has been formatted slightly.

Your code should pass all possible tests, not just the sample tests.

You should not overfit to the sample tests.

Common Mistakes

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

int main(){
    string S;
    cin >> S;
    if (S == "rose"){
        cout << "red";
    }else if (S == "violet"){
        cout << "blue";
    }else{
        cout << "sommon you";
    }
    return 0;
}
```

Note: The code has been formatted slightly.

Beware of typos!

If you need to deal with strings, copy them from the problem statement directly.

Crab Me Crab

TS2602

Problem Statement

There are two integers A and B.

Find the integer position(s) that minimises the maximum distance between it and A and B

→ the position(s) closest to the midpoint of A and B

Why?

If you move the point towards one of the original, the distance to the other will increase.

→ the maximum distance will increase

→ the midpoint is better than this new point

Therefore, the midpoint is better than any other point and is the best.

Fun Fact

As you might notice later, this problem might be accidentally solved very easily!

C++ division helps you a lot in this problem.

There was a heated discussion on the subtasks of the problem due to that fact.

Subtask 1

$0 \leq A \leq 1000$

$B = 0$

The value of B is set.

You only need one variable to keep track of A.

How to find the midpoint of A and 0?

$\frac{A}{2}$ is the solution!

Subtask 1

When A is even, there is no problem and you can just output $\frac{A}{2}$.

When A is odd, $\frac{A}{2}$ is no longer an integer!

Both $\left\lfloor \frac{A}{2} \right\rfloor$ and $\left\lceil \frac{A}{2} \right\rceil$ work now as the actual midpoint is just between them.

For example, take $A = 5$ and $B = 0$

$\left\lfloor \frac{5}{2} \right\rfloor = 2$ and $\left\lceil \frac{5}{2} \right\rceil = 3$

The actual midpoint is $\frac{1+4}{2} = 2.5$.

Since the distance between 2.5 and both 2 and 3 is 0.5, they are both valid answers

Subtask 1

However, you can not think about this and accidentally solve it!

Why?

C++ division rounds down automatically!

So, if you just output $\frac{A}{2}$, C++ will reward you with points!

Full Solution

You need two variables to keep track of A and B now.

We can do the following to find the midpoint:

Let C be the midpoint of A and B.

Without loss of generality, assume $A < B$.

$$C - A = B - C$$
$$2C = A + B$$
$$C = \frac{A + B}{2}$$

Actually, the math is useless. You can easily observe it without math.

Full Solution

As you might notice, $\frac{A+B}{2}$ might not always be an integer.

We can split the problem into 2 cases to deal with that..

Afterwards, we can combine the solution to get a full solution!

Full Solution

Case 1: $A \equiv B \ mod \ 2$

In this case, the answer is just simply $\frac{A+B}{2}$ as the midpoint directly lies on that position.

Case 2: $A \not\equiv B \ mod \ 2$

Both $\left\lfloor \frac{A+B}{2} \right\rfloor$ and $\left\lceil \frac{A+B}{2} \right\rceil$ work as the actual midpoint is just between them.

Hence, we can just use for the first case $\left\lfloor \frac{A+B}{2} \right\rfloor$ and implement it normally.

Let C++ handle the rest just like subtask 1!

Remember to use `long long`!

Subtask 2

$0 \leq A, B \leq 1000$

A, B are even

A safety net in case you somehow use $\frac{A}{2} + \frac{B}{2}$ instead of $\frac{A+B}{2}$.

Who would even do that???

What was the problem author doing when they were thinking about this subtask???

The author of this editorial after the contest: See? Not a single person, not even a single submission died here (very much expected).

Subtask 3

$0 \leq A, B \leq 1000$

A safety net in case you forgot `long long` and used `int`.

Take-home Task

If you need to output all solutions, how can you do that?

Hint: Use if-statement to differentiate between the two cases and use some math to calculate $\left\lceil \frac{A+B}{2} \right\rceil$.

Takeaway

`long long` is very commonly needed in OI problems.

The problem setters were nice and gave a lot of pity points to the people who forgot. This might not be the case next time.

Remember to use `long long`.

Remember to use `long long`.

Remember to use `long long`.

Very important things say three times!

Reading problem constraints is a good habit too.

Common Mistakes

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

int main(){
    int A, B, sum;
    cin >> A >> B;
    sum = (A + B) / 2;
    cout << sum;
}
```

Note: The code has been formatted slightly.

Use long long, results go strong!

Die on long long, many marks gone!

Water Bird

TS2603

Fun Fact

一石二鳥

There is a problem called “One Stone Two Fish” (not bird!) on our online judge.

If you are interested, you can try to solve it.

Problem Statement

There are N birds, 1 out of F birds will be washed away (remaining birds all survive).

How many remain?

Subtask 1

$\mathrm{N} = F$

One out of the N birds will be washed away since $\mathrm{N} = F$!

How many remain?

The answer is obviously $\mathrm{N} - 1$!

Subtask 2

N is even
$\mathrm{F} = 2$

One out of two birds will be washed away.

So, half will remain!

The answer is $\frac{N}{2}$.

Subtask 3

$\mathrm{F} = 2$

Starting from this subtask, we have to deal with the remaining birds.

One out of two birds will be washed away.

However, there might be remaining birds that will survive.

We can do some math and find the number of birds that wash away and subtract that from N to solve this subtask.

Subtask 3

We can use the property of C++ division to find the number of birds that will be washed away. Since C++ division rounds down automatically, we can just do $\frac{N}{2}$ and it will work.

Therefore, we can output $\mathrm{N} - \frac{N}{2}$ and C++ will help us!

Full Solution

Now, we need to use two variables to get the values of N and F.

Just chain together `cin` and two sets of `>>` and we can get both values!

Now, $\left\lfloor \frac{N}{F} \right\rfloor$ will be washed away.

Like subtask 3, we can use $\frac{N}{F}$ and C++ will do the rest.

Do the subtraction and we will get the answer.

Takeaway

Math is very useful in OI. There are a lot of math problems. If you know math, it can help you solve problems way easier.

However, if you are not the best at math, don’t give up. You still can achieve great results in OI.

For example, you can simulate the process in the problem and solve it without math if the constraints are smaller. However, the math solution will be shorter and more elegant.

Using math can reduce the time complexity of your code too as most math operations are $O(1)$.

Common Mistakes

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

int main(){
    int A, B;
    cin >> A >> B;
    if ((A + B) == 10){
        cout << 4 << endl;
    }
}
```

Note: The code has been formatted slightly.

Don’t write code that solves only a part of the problem.

Also, don’t submit code that you are sure is wrong and cannot get any marks as other contestants rely on the judge to receive a verdict for their submissions. Don’t waste valuable resources.

Common Mistakes

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

int main(){
    int N, F;
    cin >> N >> F;
    if (N = F){
        cout << F - 1 << endl;
    }
}
```

Note: The code has been formatted slightly.

This person is attempting subtask 1.

However, they clearly did not pay attention during our course and used a single = sign to check for equality and therefore should not pass our test (how unfortunate)!

However, their code surprisingly complies and even works!

How come? It is left as an exercise for the readers.

pH Value

TS2604

Problem Statement

Given a decimal number with one decimal place N, output

- `acidic` if $N < 6.5$
- `neutral` if $6.5 \leq N \leq 7.4$
- `basic` if $N > 7.4$

Fun Fact

Fun fact:

In chemistry, only a pH of exactly 7 is considered neutral.

This problem is not accurate and will give you Jack Wong grades it you learn from it.

Subtask 1

$N = 7.0$

Read the problem statement.

Since $6.5 \leq 7.0 \leq 7.4$, you know what to do.

Subtask 2

$0.0 \leq N \leq 7.0$

A simple if-else will do the trick.

Fun fact: `string` can successfully solve this subtask too!

Why?

It is left as an exercise for the reader.

Full Solution

We now have to deal with all 3 cases.

Just like in Rose and Violet, we can use `if`, `else if`, `else`.

But how can we check for $6.5 \leq N \leq 7.4$?

Use `&&` to check if N satisfies both $N \geq 6.5$ and $N \leq 7.4$.

Alternatively, we can put the $6.5 \leq N \leq 7.5$ in the `else` branch and no logic operators will be required.

Oh no! Unfortunately, `float` somehow can not solve this problem. However, it is common practice that we use `double` in OI. Therefore, we hope that you got your points.

Subtask 3

$N = x.0$, where $0 \leq x \leq 14$

A safety net in case you only know how to use `int` and not `double`.

Or maybe you don't know how to implement the check for $6.5 \leq N \leq 7.4$ and can only check for $N = 7$.

Alternative Solution

Use `round` if you know how to.

Then you can avoid the need of using logic operators (although you can avoid it another way too).

Takeaway

Beware of floating-point inaccuracies. They will appear a lot if you are using them.

Most problems can be solved using integers only, and you only need to do a floating-point operation at the end to minimise floating-point errors.

For most problems, use `double` or `long double` instead of `float`.

Takeaway

Sometimes, there are more than one methods of solving a problem. Try to find a simpler method or one that you are confident in implementing.

If you don’t how to implement a solution, try to find another way to do so instead of giving up. There might be workarounds.

The C++ Standard Library might contain useful functions to help you write better code too.

You can learn some common useful features from writing more code and looking at others demonstrate more.

Common Mistakes

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

int main(){
    double N;
    cin >> N;
    if (N < 6.5){
        cout << "acidic";
    }else if (N > 7.5){
        cout << "basic";
    }else if (N <= 7.4){
        cout << "neutral";
    }
    return 0;
}
```

Note: The code has been formatted slightly.

Off-by-one errors are common!

A pH value of 7.5 is also basic!

Change the `N > 7.5` to `N >= 7.5`.

Common Mistakes

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

int main(){
    long double N;
    cin >> N;
    if (N >= 6.5 && N <= 7.4){
        cout << "neutral";
    }
    if (N >= 7.4){
        cout << "basic";
    }
    if (N <= 6.5){
        cout << "acidic";
    }
}
```

Note: The code has been formatted slightly.

Off-by-one errors are common (part 2)!

If `N == 6.5` or `N == 7.4`, bad things will happen!

Common Mistakes

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

int main(){
    int N;
    cin >> N;
    if (N >= 6.5 && N <= 7.4){
        cout << "neutral";
    }else if (N > 7.4){
        cout << "basic";
    }else if (N < 6.5){
        cout << "acidic";
    }
}
```

Note: The code has been formatted slightly.

`int` cannot store decimal numbers.

Use `double` or `long double`!

Zodiac Animals

TS2605

Problem Statement

Output the corresponding zodiac animal of year Y.

(`Rat`, `Ox`, `Tiger`, `Rabbit`, `Dragon`, `Snake`, `Horse`, `Goat`, `Monkey`, `Rooster`, `Dog`, `Pig`, `Monkey`)

Fun Fact

The problem constraint for this problem was one very small!

However, we feared that a student M will spend the whole 2:30 using if-statements to solve it.

So, the constraint on Y was increased to prevent that.

Also, for those who know, HOLY GD REFERENCE!!!

Subtask 1

$2025 \leq Y \leq 2026$

If $Y = 2025$, output `Snake`, else output `Horse`.

A simple if statement can help you.

Subtask 2

$2025 \leq Y \leq 2031$

Now, there are 12 cases. Starting from 2025, we need to output `Snake`, `Horse`, `Goat`, `Monkey`, `Rooster`, `Dog`, `Pig`, `Monkey`, `Rat`, `Ox`, `Tiger`, `Rabbit`, `Dragon` all the way to 2031.

Subtask 2

If you want to, you can still make a huge if statement with 12 branches to solve this subtask.

Does it work? Yes!

Is it dumb? Yes!

When making 12 branches, we might make a mistake and spend a lot of time debugging (and even not being able to solve this subtask)!

To save our precious time in contests, how can we avoid writing so many lines of code?

Array!

Subtask 2

We can create an array of string storing all the zodiac animals.

Then, when we need to output the answer, we can convert the year into a suitable index for the array and get the answer easily.

Full Solution

Now, Y can be very large.

The solution from subtask 2 doesn't work anymore as the index can become very large and exceed the array length.

How can we deal with that?

We can write thousands of lines of code to solve it using if statement!

Ok, just kidding. You don't want to be another student M.

Full Solution

Remember the humble `%`? It takes the modulo of a number!

How can that help us?

The remainder of increasing integers effectively cycles.

e.g. the remainder of integers after diving by 12 starting from 0

$0 \rightarrow 1 \rightarrow 2 \rightarrow \dots \rightarrow 10 \rightarrow 11 \rightarrow 0 \rightarrow 1 \rightarrow 2 \rightarrow \dots \rightarrow 10 \rightarrow 11 \rightarrow \dots$

Again, how can that help us?

Full Solution

Just like in the system of zodiac animals, the remainders “roll back” and continue cycling.

Therefore, we can utilise the power of modulo 12 and use that to convert the year into a suitable index.

After that, we can reference the array and get our answer.

Indeed, we can still use if statements with `%` to find the answer or even use other tools such as looping to achieve the same result. However, this method is still the cleanest and less bug-prone.

Takeaway

In OI contests, we should implement methods that are simpler and less error-prone to avoid the need of debugging.

When you find that you are writing a lot of code to solve a simple problem, you should stop and ask yourself is there another method that is better than your current one.

The problem setters *most likely* won't require you to write 691 lines of code to AC a problem.

By cleverly thinking about how to write more concise code, you will save yourself from a lot of headaches later.

Common Mistakes

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

int main(){
    long long a;
    cin >> a;
    if (a % 12 == 4){
        cout << "rat" << endl;
    }else if (a % 12 == 5){
        cout << "Ox" << endl;
    }else if (a % 12 == 6){
        cout << "Tiger" << endl;
    }else if (a % 12 == 7){
        cout << "Rabbit" << endl;
    }else if (a % 12 == 8){
        cout << "Dragon" << endl;
    }else if (a % 12 == 9){
        cout << "Snake" << endl;
    }else if (a % 12 == 10){
        cout << "Horse" << endl;
    }else if (a % 12 == 11){
        cout << "Goat" << endl;
    }else if (a % 12 == 0){
        cout << "Monkey" << endl;
    }else if (a % 12 == 1){
        cout << "Rooster" << endl;
    }else if (a % 12 == 2){
        cout << "Dog" << endl;
    }else if (a % 12 == 3){
        cout << "Pig" << endl;
    }
    return 0;
}
```

Note: The code has been formatted slightly.

Outputs are case-sensitive in most cases.

`Rat` should be outputted instead of `rat`.

Yet again, just copy and paste.

Don’t type.

Common Mistakes

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

int main(){
    int y, a, x = 2020;
    while (cin >> y){
        a = (y - 2020 + 1) % 12;
        if (a == 1){
            cout << "Rat" << endl;
        }else if (a == 2){
            cout << "Ox" << endl;
        }else if (a == 3){
            cout << "Tiger" << endl;
        }else if (a == 4){
            cout << "Rabbit" << endl;
        }else if (a == 5){
            cout << "Dragon" << endl;
        }else if (a == 6){
            cout << "Snake" << endl;
        }else if (a == 7){
            cout << "Horse" << endl;
        }else if (a == 8){
            cout << "Goat" << endl;
        }else if (a == 9){
            cout << "Monkey" << endl;
        }else if (a == 10){
            cout << "Rooster" << endl;
        }else if (a == 11){
            cout << "Dog" << endl;
        }else if (a == 12){
            cout << "Pig" << endl;
        }
    }
}
```

Note: The code has been formatted slightly.

When a number is divisible by 12, the remainder modulo 12 is 0, not 12.

In addition, we don't need to combine `while` with `cin` as there is only one integer in the input.

Five O Five

TS261

Problem Statement

Given a time in p.m., output the difference in minutes of that time with 5:05 p.m.

Fun Fact

There is a game in OI team where we guess who made the 5:05 announcement.

There is a sign just outside of the hall that shows the teacher who made the announcement.

Subtask 1

$H = 5$
$6 \leq M \leq 59$

Hmm... Huh??? How to do???

If you somehow have learnt math before, you will quickly realise that the answer is just $M - 5$!

Why?

Think yourself.

Subtask 2

$H = 5$

There are 2 cases.

Either the time is earlier than 5:05 p.m., or later.

If the time is later than 5:05 p.m., subtask 1.

Else, just subtract M from 5.

Actually, there are 3 cases.

But the remaining one is trivial.

Either way will produce the right answer.

Subtask 3

$1 \leq H \leq 4$

The time must be earlier than 5:05 p.m.

Subtract H from 5, and multiply by 60.

That will be the difference of the H:05 p.m. and 5:05 p.m. time.

Use the method from subtask 2 to subtract/add on the minute-difference from the reference point of H:05 p.m.

Subtask 4

$6 \leq H \leq 11$

The time must be later than 5:05 p.m.

Subtract 5 from H, and multiply by 60.

You will find the H:05 p.m. time.

Use the method from subtask 2 to subtract/add on the minute-difference from the reference point of H:05 p.m.

The only difference between this and subtask 3 is the order of subtraction.

Subtask 5

$M = 5$

We don’t need to deal with minute calculations in this subtask!

Just find the difference between H and 5, and multiply by 60.

Make sure you check if H is smaller than or bigger than 5.

Subtask 5

I did what you told me to, but why can't I solve it???

Unfortunately, $H = 12$ is a special edge case!

Do you remember that 12 p.m. is actually in the afternoon, and not midnight?

It actually behaves like 0 p.m.

Just add one more branch, and you will be done!

This subtask was also designed for people to find out what is wrong with their full solution code when $H = 12$.

Full Solution

Combine all the ideas of the subtasks:

- Hour and minute difference calculations.
- 12 p.m. edge case.

You will be able to solve this problem easily!

Full Solution

You might notice that the code is quite long and complicated.

We can apply these tricks to simplify our code:

Remember our good friend `%`?

The time system is just like a modulus system!

We can use $H \% 12$ in our calculations and save the extra branch.

We can use `abs` for finding the difference. It returns the non-negative magnitude of a number (i.e. the distance of that number and 0), perfect for finding the difference which is always positive.

Just do the subtraction and use it to save us from needing to use if-statement.

Full Solution

Alternatively, you can think of it as a timestamp.

5:05 p.m. is some minutes after 12:00 p.m.

H:M is also some minutes after 12:00 p.m.

By thinking of it this way, we can find the minute difference easily.

The same tricks can be made to make the code for this method simpler.

Take-home Task

If there is an extra character in the input C, which can be `A` or `P`, representing a.m. or p.m., how can you solve this problem, given that the time given and the actual 5:05 p.m. is on the same day?

Hint: a.m. is just another minute difference to be added compared to p.m.

Takeaway

Work smarter, not harder.

Can I write 12×60 branches of an if-statement? Sure!!!

Do you want to do that? Probably not, unless you are student M.

With clever thinking, we can separate the cases into a few big group, and in this case, combine them into one with some final tricks.

Some problems have similar methods to make code simpler. If you can think of them, definitely use them instead of following student M!

Common Mistakes

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

int main(){
    long long H, M;
    cin >> H >> M;
    if (H = 5 && M >= 5){
        cout << M - 5 << endl;
    }else if (H = 5 && M <= 5){
        cout << 5 - M << endl;
    }
}
```

Note: The code has been formatted slightly.

Again, this person clearly did not pay attention during our course and used a single = sign to check for equality.

However, they still successfully got in! We should have failed them by now!

However, their code surprisingly complies and even works for the intended subtasks!

How come? It is left as an exercise for the readers.

Common Mistakes

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

int main(){
    int h, m;
    cin >> h >> m;
    int oh = 5, om = 5;
    cin >> oh >> om;
    int hour = h - oh;
    int min = m - om;
    int ans = 0;
    if (h != 12){
        ans += hour * 60 + min;
    }else if (h == 12 && m == 0){
        hour = 5;
        ans += hour * 60 + min;
    }else if (h == 12 && m > 0){
        hour = 4;
        min = 60 - m + om;
        ans += hour * 60 + min;
    }
    if (ans < 0){
        ans = ans - ans - ans;
    }
    cout << ans;
}
```

Note: The code has been formatted slightly.

This is a good attempt at a full solution, separating into different cases.

Unfortunately, there is a bug in the $12:00$ branch.

Debug that branch, and you can AC!

You can even AC by removing that branch and just using the last one too!

Even if you are wrong only once, you will still WA...

ConteStats

TS262

Problem Statement

Given N integers, output their mean (rounded down), minimum and maximum. (so short!)

Fun Fact

The problem author was considering adding more different statistics in the problem (e.g. median, standard deviation), but these require more advanced techniques.

Hence, they were removed to keep the problem simple. (Good for all of you Team Selection Test takers!)

Subtask 1

$S_1 = S_2 = \cdots = S_N$

Every number in the data set is the same!

What does that imply?

The mean, minimum and maximum will all be the same (any one of the N integers)!

Just input S_1, output it three times and you are good to go.

Subtask 2

$S_i = i$

How can we find the mean, minimum and maximum in this case?

We need to do some thinking.

We can separate the problem into two parts, finding the minimum/maximum and the mean.

Subtask 2

Finding the minimum/maximum:

This is easy to do!

Out of all the numbers from 1 to N, the minimum and maximum is obviously 1 and N respectivety!

Subtask 2

Finding the mean:

We can observe that the mean is just right in the middle of 1 and N. Hence, it is $\frac{(N+1)}{2}$.

Just output it and you can get the remaining 50% of the points from subtask 2.

Full Solution

Oh no! The integers are not set anymore! We can't do some clever thinking to work out the answer!

How can we get N integers?

May I introduce:

The for-loop!

We can loop N times to get the integers.

Now, we just have to find the minimum/maximum and the mean.

Full Solution

Finding the mean:

We can just add all the integers up and divide the result by N.

Since C++ division rounds down automatically (huh where have I seen this fact before?), we don't need to do any extra work!

Use `long long` as adding up the integers will cause overflow.

Full Solution

Finding the minimum/maximum:

We can loop and use the `min`/`max` functions to calculate it.

If you didn't pay attention in the C++ Foundation Course and don't know about them, we can still use if-statement to achieve the same result.

The problem author was extra nice as even if you forget `long long`, you still can get this 50%.

Alternative Solutions

1. Use C++ `sort` and retrieve the first and last element for the minimum and maximum.

However, this is O($NlgN$).

2. Use C++ `min_element`, `max_element` and `accumulate`. If you want to know how to use them, refer to cppreference.

Subtask 3

$1 \leq S_i \leq 10^4$

A safety net in case you forgot `long long`.

Make sure that it won't happen again.

Takeaway

Yet again, if you forgot about a C++ function, there are most likely workarounds that don’t require them.

Even though implementing them takes up more time and is more bug-prone, if you can’t think of the function, you should still do it. (getting points is better than getting no points!)

Takeaway

Partial scoring is useful and can help you get points if you can’t fully solve the problem.

You should read the scoring rules carefully and plan on getting the maximum score possible.

Sometimes, they might give you an insight on how to solve the problem.

Due to checker constraints, you might need to output garbage values to match the output format of the problem to receive the partial scoring. Keep that in mind!

However, for this time, we had to write the checker very carefully to make people who output wrongly wrong.

Common Mistakes

Note: The code has been formatted slightly.

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

int main(){
    int n, s[200111], ans = 0;
    cin >> n;
    for (int i = 0; i < n; i++){
        cin >> s[i];
        ans += s[i];
    }
    sort(s, s + n);
    cout << s[0] << " " << s[n - 1] << "\n";
    cout << ans / n;
}
```

Use `long long`, results go strong!

Die on `long long`, many marks gone!

Common Mistakes

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

int main(){
    long long N, s, M = 0, m = 10000, a = 0, ans;
    cin >> N;
    for (int i = 0; i < N; i++){
        cin >> s;
        a = a + s;
        if (s > M){
            M = s;
        }
        if (s < m){
            m = s;
        }
    }
    ans = a / N;
    cout << m << ' ' << M << endl;
    cout << ans;
    return 0;
}
```

Note: The code has been formatted slightly.

The minimum value can be very large.

However, it was bounded at 10000.

Even if the minimum value is larger than 10000, the code still outputs 10000.

Work Past Password

TS263

Problem Statement

Output the password which loops starting with character S and ending with character C.

Fun Fact

We can use if-statement to solve this problem.

We can convert the current character to the next one in just 26 cases!

Now, student M is probably solving this problem using this method because long code is his favourite!

Subtask 1

$S = \texttt{a}$
$N = 3$

There are only 2 possibilities for the resulting password.

If $E = \texttt{B}$, the password is `aba`.

Otherwise, the password doesn't loop back and is `abc`.

Subtask 2

The pattern consists of distinct characters.
i.e. the pattern will not repeat.

In this subtask, we should use character operations.

Refer to the 2025/26 C++ Foundation Course Chapter 2 for more details.

Now, we can repeatedly add 1 to the starting character until it is the ending character.

You can self-code this or just use for-loop.

Subtask 3

$S = \texttt{a}$

$E = \texttt{b}$

The pattern will look like `abab`...

Just loop and output `a` and `b` alternately.

Subtask 4

E is immediately followed by S in the alphabetical order

Then the pattern will look like $ESES$...

Just loop and output E and S alternately.

(huh why so similar to subtask 3?)

Subtask 5

$S =$ a

$E =$ z

We will loop through all letters from a to z.

However, as hardcoding this is quite annoying (except for student M), we need to use a smarter method.

Remember character operations?

We can use it to loop from a to z.

Loop N times, take the index $\% 26$ and add it to the character 'a', and output it.

Full Solution

For the full solution, we can do the following:

Starting from character S, you can add add 1 to the character.

If it equals E, you need to set it back to S.

Just rinse and repeat N times!

Easy!

Takeaway

C++ data types are very versatile.

If you know them well, you can save a lot of unnecessary work.

Pay attention in later training classes to learn them properly!

Common Mistakes

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

int main(){
    char S, E, sav;
    int N;
    cin >> S >> E >> N;
    sav = S;
    for (int i = 0; i < N; i++){
        cout << S;
        if (S == E){
            S = sav;
        }else{
            S++;
        }
    }
    cout << endl << sav;
    return 0;
}
```

Note: The code has been formatted slightly.

An extra debug message was outputted.

Even if your code is correct, you will still receive a Wrong Answer!

CapitaliStatement

TS254

Problem Statement

Change the cases in the string S to follow the following rules:

- The first letter should be in uppercase if it is at the start of a sentence, otherwise, it should be lowercase
- All other letters should be in lowercase

Fun Fact

The last time a problem similar to this was proposed, there was an extremely simple python solution! (a built-in method that returns the answer)

This year, the problem was proposed way better, and the proposer actually knows python!

Therefore, this problem has no built-in solution in python.

By the way, if you are brave enough, you can skip a key idea in this problem and use a long if-statement the student M way.

KEY IDEA ONE

Input With Spaces

Normally, we use `cin >> S` to input a string.

However, it will stop once it encounters a space.

How to fix that?

We can use `getline(cin, S)`!

When we use that function, it will input the whole line into our string, instead of stopping at the first space.

KEY IDEA TWO

First Letter Identification

How can we know if a character in our string is the first letter of the sentence?

The first character obviously is the start of a sentence, but how about others?

A sentence ends with `.`, `?` or `!`.

We can use this to detect whether a letter is the first of a sentence!

We can store whether the last character of the last word is a punctuation mark or not.

Note that the last character of the last word is before the space in front of the current word (i.e. two characters back), so be careful and don't access out of bounds.

Now, we just have to use that information to decide what we should do.

KEY IDEA ONE + TWO

Input With Spaces + First Letter Identification

Now, we can combine both ideas.

Remember that we can use `getline` and some careful checking to do the job?

However, that method is slightly more complicated than necessary. (but it still solves this problem, so you can use it)

Instead, we can use `while` combined with `cin` to input one word at a time!

The syntax is `while (cin >> S)`.

Now, we can just directly check the last character of the last word, by using a Boolean flag for example.

KEY IDEA THREE

Letter Case Modification

Now, given that we know which letters should be turned into uppercase, and which should be turned into lowercase, how to actually make the modification?

Easy!

Just do character addition and subtraction.

Make sure that you check whether the original case is already correct. You don’t want “upperestcase” or “lowerest” case!

What I am going to present to you is basically a repeat of 2024/25 C++ Foundation Course slides Chapter 8.

Refer to it if you want to.

KEY IDEA THREE (I)

Letter Case Modification (Checking)

For lowercase to uppercase, we can use `'a' <= S[i] && S[i] <= 'z'`.

For uppercase to lowercase, we can use `'A' <= S[i] && S[i] <= 'Z'`.

Use these to conditions in the if-statement surrounding the case changing.

KEY IDEA THREE (II)

Letter Case Modification (Changing)

For lowercase to uppercase, we can use `S[i] += 'A' - ‘a'`.

For uppercase to lowercase, we can use `S[i] += 'a' - 'A'`.

Use an if-statement to surround these.

Subtasks Overview

Subtask 1 (`A`, `a` and `.`): You can simply output `Aaaa...aaa.` (length of $S - 2$ `a`'s).

Subtask 2 (lowercase letters, `.`, `!` and `?`): Use key idea 3 (I) to change characters 2 to length of $S - 1$ into uppercase.

Subtask 3 (letters, `.`, `?` and `!`): Key idea 3 (II) is used here. Build upon the solution for subtask 2. Handle the first letter carefully too.

Subtask 4 (`A`, `a`, `.`, `?`, `!` and spaces): Use key ideas 1 and 2 and hardcode the uppercase and lowercase of `a`.

Subtask 5 (lowercase letters, `.`, `?`, `!` and spaces): All ideas except key idea 3 (II) are used.

Full Solution

We first input the sentence using key idea 1.

For each word in the sentence, we check if the first letter is the start of a new sentence using key idea 2.

Then, we modify the case accordingly using key idea 3.

Combine all these ideas, and you will solve this problem easily!

Takeaway

Most problems consists of parts and ideas that you need to use and combine to write a full solution.

You should try to think clearly before writing any code.

Sometimes, subtasks can hint to the key ideas.

Common Mistakes

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

int main(){
    string s;
    int i = 1;
    cin >> s;
    for (int j = 0; j < s.size(); j++){
        if (i == 1){
            if (s[j] >= 'a' && s[j] <= 'z'){
                cout << (char)(s[j] - 32);
            }else{
                cout << s[j];
            }
            i = 0;
        }else{
            if (s[j] == '.' || s[j] == '?' || s[j] == '!'){
                cout << s[j];
                i = 1;
            }else if (s[j] == ' '){
                cout << " ";
            }else{
                if (s[j] >= 'A' && s[j] <= 'Z'){
                    cout << (char)(s[j] + 32);
                }else{
                    cout << s[j];
                }
            }
        }
    }
}
```

Note: The code has been formatted slightly.

There was no space handling in this code.

We need to check for spaces and use `getline`.

Window Installer

TS265

Problem Statement

Just read and understand it.

Fun Fact

As you can see from the last slide and the latter subtasks, the problem was once considered poorly designed, and we wanted to remove it from the test.

However, as this problem is considered simple, we kept it so that more people could get more point (how nice of us!).

Subtask 1

$N = 3$

Just output the following:

```
###
#+#
###
```

You can make it yourself, or better yet just copy from the sample tests!

Subtask 2

$N = 5$

Just output the following:

```
#####
# | #
#-+-#
# | #
#####
```

This time, you can’t copy and need to create it yourself.

Subtask 3

$N = 7$

Just output the following:

```
#######
#  |  #
#  |  #
#--+--#
#  |  #
#  |  #
#######
```

You can make it yourself, or better yet just copy from the sample tests!

Huh, Déjà vu!

Expected Score	9	Cumulative	19

Subtask 4

$3 \leq N \leq 9$

Create the window for the 9×9 case and use if-statement and the window from the other subtasks to solve this subtask.

Full Solution

To solve this problem, we need a tool.

The tool is nested for-loop!

Just loop $N \times N$ times and output the characters according to the indices.

Use if-statements combined with for-loop indices to see if you should output `#`, `-`, `|`, `+` or ` `.

Takeaway

The problem setters are nice sometimes and can give you marks for free!

Look at the sample tests and see if you can use them.

Even if they can’t give you free points, you still should use them to understand the problem better and test your code! After all, that’s what “samples” are for.

~~For your information, not even a single person who wasn’t able to solve this problem copied the sample tests to submit. No one got free points!~~

Well, this is not true anymore. One person did do it. Only **ONE**.

Common Mistakes

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

int main(){
    char w[100][100];
    int N;
    cin >> N;
    for (int i = 0; i < N; i++){
        w[0][i] = '#';
        w[i][0] = '#';
        w[N - 1][i] = '#';
        w[i][N - 1] = '#';
    }
    for (int l = 1; l < N - 1; l++){
        w[(N - 1) / 2][l] = '-';
        w[l][(N - 1) / 2] = '|';
    }
    w[(N - 1) / 2][(N - 1) / 2] = '+';
    for (int j = 0; j < N; j++){
        for (int k = 0; k < N; k++){
            cout << w[j][k];
        }
        cout << endl;
    }
}
```

Note: The code has been formatted slightly.

The array was not initialised.

Therefore, the code outputs incorrectly when it should output space.

Love Leafer

TS266

Problem Statement

How many tree blocks have one and only one orthogonal tree block neighbour?

Fun Fact

This problem was created as I was making a lot of small optimisations on Leaf Lover and needed a way to judge their effectiveness. I used it as a checker for my code.

The problem author (me!) has the highest score on Leaf Lover and is conjectured to be the maximum possible.

I also considered adding the check for connectivity in this problem, however, as that requires more advanced skills, the idea was scrapped.

The name Love Leafer came from Felix misspeaking Leaf Lover.

Subtask 1

$N = 1$
$M = 2$

There are only 4 cases.

Use if-statement and you are good to go.

If you are smarter, you might realise that all cases except `##` have no leaves.

Then you can just use one simple if-else to do it.

Subtask 2

$N = M = 2$

There are only 16 cases.

Use if-statement and you are good to go.

However, you will not write so many branches normally (unless you are student M).

We can group the cases to make our code simpler.

There are a lot of different ways to do so.

Try and figure out one yourself!

Input the grid carefully.

Subtask 3

$N = 1$

We need to use loops to do this problem now.

Just loop through the 1D grid and count the number of tree blocks that have only a left or right tree block.

Beware of edge cases that are at the *edge* of the grid.

You need to check and deal with them.

Subtask 4

Each `#` has exactly 1 adjacent `#`

If each tree block has exactly 1 neighbour tree block, what does that mean?

All of the tree blocks are leaves!

Just count the number of `#` and you can solve this subtask easily.

Subtask 5

There are exactly 2 `#` among $T_{i,j}$.

If there are only 2 tree blocks, we can just see if they are neighbours. If they are, there are 2 leaves. If they aren't, there are none.

How can we find that?

Just save the coordinates of the tree blocks. Afterwards, check if only the x or y coordinate differ by 1 (not both).

Full Solution

Now, a 2D array of characters or a 1D array of string is required.

Nested for-loops are required too.

For each cell, check if it is a tree block. If it is, check if it has exactly one neighbour tree cell. If it is, add one to the total leaf count.

We can use helper arrays and another layer of nested for-loop to help us avoid the need of a lot of if-statements.

Full Solution

We can create 2 arrays for the offsets of the x and y coordinates for all 4 orthogonal directions (up, down, left, right).

Loop 4 times and add the offset to the current coordinate for the new coordinate.

If the coordinate goes out of bounds, skip it. Otherwise, check if it is a tree block.

Afterwards, just update the counter accordingly.

Full Solution

When all of that is finished, output the total leaf count and you are done!

Yet again, be careful while inputting the grid.

Subtask 6

$T_{1,j} = T_{N,j} = T_{i,1} = T_{i,m} = .$

A safety net in case your code doesn't handle the edges properly.

However, when your code accesses uninitialized or out of bounds values, it might still get a very good score or even AC! Since the values there are random, the chances of them being something that will make you WA are very low. Hence, there was a large number of rejudges for this problem.

Take-home Task

If you need to output `0` if some tree blocks are not connected, how can you do that?

Hint: You can use BFS or DFS. If you want to know what those are, get a medal in HKOI and join the training team. :)

Takeaway

Handle edge cases carefully. If you don’t handle them, bad things will happen.

In this case, the you might accidentally “AC” even if your code is wrong.

However, that won’t happen normally. You might even receive a 0 if the problem setters are not nice.

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

int main(){
    char t[111][111];
    int n, m, ans = 0;
    cin >> n >> m;
    for (int i = 1; i <= n; i++){
        for (int j = 1; j <= m; j++){
            cin >> t[i][j];
        }
    }
    for (int i = 1; i <= n; i++){
        for (int j = 1; j <= m; j++){
            if (t[i][j] == '#'){
                int l = 0;
                if (t[i - 1][j] == '#'){
                    l++;
                }
                if (t[i + 1][j] == '#'){
                    l++;
                }
                if (t[i][j - 1] == '#'){
                    l++;
                }
                if (t[i][j + 1] == '#'){
                    l++;
                }
                if (l == 1){
                    ans++;
                }
            }
        }
    }
    cout << ans;
}
```

Common Mistakes

Note: The code has been formatted slightly.

The array was not initialised.

Therefore, the calculations may wrong.

However, as it is quite rare for it to be uninitialized such that it is incorrect. Therefore, this code passed.

As we felt extra generous, we didn’t change the score.

Judge Addict

TS266

Problem Statement

Count how times does each number appear. If it doesn’t appear at all, don’t output it.

Fun Fact

The subtasks for this problem are yet again, not good.

This problem is way easier than its placement in our test in my opinion.

Subtask 1

$N = 2$

There are only 2 cases.

Case 1: Both records are from the same user.

Just output the id and `2`.

Case 2: The records are different users.

Just output the smaller id and `1`, and the bigger id and `1` (so repetitive!).

Subtask 2

$N = 3$

It is very similar to the last subtask, but more cases.

There might be 0 to 3 records that are from the same user.

Remember to output them in the correct order.

Subtask 3

$1 \leq U_i \leq 2$

There are exactly 2 users who have logged in at least once.

Just check if each number is 1 or 2.

Then output the number of 1s and 2s.

Subtask 4

$1 \leq U_i \leq 100$

There are exactly 100 users who have logged in at least once.

For each user, loop through all the records and count how many records are from that user.

Who would even do this subtask instead of full solution???

The author of this editorial after the contest: See? Not a single person, not even a single submission did this subtask only instead of the full solution.

Subtask 5

$1 \leq U_i \leq 100$

This subtask is built upon the last one.

Now, we have to output the counts carefully. If it is 0, we should not output it.

Use if-statement to check it.

Fun fact: You can just do `if (arr[i])` as C++ will auto-cast to Boolean and all non-zero values will return `true`.

O ($Nmax(U_i)$)	Expected Score	**42**	Cumulative 75

Full Solution

You will need to declare an array. For every record, increment the correct value of the array.

Then just output all of them at the end. Don't output them if it is 0.

Alterative Solution

Indeed, you can use map to solve this problem.

If you want to learn more about map, refer to the 2026 C++ Foundation Course Chapter 14.

If there are only 2×10^5 users, array will be the better option.

However, if there are a large number of users (e.g. 10^9), map will definitely be the better choice as it avoids exceeding the memory limit.

The memory complexity of the normal solution is O($\max(U_i)$) while this one is O(N).

Takeaway

This the the basic implementation of something called frequency array.

This skill might be required as a part in other problems.

You should (need to) know how to write this.

Common Mistakes

```cpp
#include <bits/stdc++.h>
using namespace std;
int arr[200000];
set<int> s;
int main(){
    int k, a, b, x;
    cin >> k;
    for (int i = 0; i < k; i++){
        cin >> x;
        s.insert(x);
        arr[x]++;
    }
    for (int i = 1; i < 200000; i++){
        if (arr[i] > 0){
            cout << i << ' ' << arr[i] << endl;
        }
    }
}
```

Note: The code has been formatted slightly.

Off by one error!

The maximum value for U_i is 200000.

However, an array of size 200000 is 0-indexed and the indices will only go up to 199999.

The `i < 200000` condition in the loop is also wrong.

The `set` is also useless.

Mark Six

TS268

Problem Statement

There are 6 drawn numbers and 1 extra number.

Given your bet and the prize money for the 7 types of prize, output the amount that you can win.

Fun Fact

Jack Wong is legally allowed to buy lottery tickets! ~~After all, he is gambling.~~

Subtask 1

$D_i = i$
$E = 7$
$P_2 = P_3 = \cdots = P_7 = 0$

You will only receive a prize when all numbers match from $1 - 6$!

Therefore, output P_1 when that happens. Otherwise, output 0.

There are many ways to check for this. Think of your own!

Subtask 2

$B_i = D_i$ for $1 \leq i \leq 5$

The first five numbers are a guaranteed match.

That leaves three cases for the sixth number.

Case 1: It matches the drawn sixth number.

It is the first prize for this case.

Case 2: It matches the extra number.

It is the second prize for this case.

Case 3: It doesn't match either of them.

It is the third prize for this case.

Observation

The match to prize conversion can be done using if-else statements.

However, that leads to long and hard to debug code.

We can notice that we can extend the prize table to include first to thirteenth prizes.

Then, each non-extra match moves the prize up two spots while an extra math moves it up one spot.

With that system in place, we can find the according prize easily!

Subtask 3

$D_i = i$
$E = 7$
$P_i = 13 - i$

This subtask somehow uses the observation but in a simplified way.

For each number, if it is between 1 and 6, add 2 to the prize total. If it is 7, add 1 to the prize total.

Output the total. (It is already the payout!)

Subtask 4

$D_i = i$
$E = 7$

Building upon the last subtask, we can calculate $13 - prize$ easily.

Just subtract that from 13, use it as an index for the prize payout, and you are done!

Remember to account for 0-base indexing.

Subtask 5

$P_2 = P_3 = \cdots = P_7 = 0$

Check if all the numbers match the drawn numbers.

If yes, output P_1. Otherwise, output 0.

Again, there are a few methods to do it.

Subtask 6

$B_i \neq E$
$P_i = 13 - i$

Again, building upon subtask 3, we can notice that we just need to handle the checking part. For each number, check if it is a drawn number. If it is, add 2 to the prize total.

Output the total.

Subtask 7

$P_i = 13 - i$

Building upon the last subtask, we can adjust the method. We need to handle the extra number.

If a number matches the extra number, add 1 to the total.

Everything else stays the same.

Full Solution

Combine the ideas of the subtasks to solve this problem.

Use the same method to find the prize index.

Then use the same method to find the payout amount.

Easy!

Takeaway

There are indeed a lot of ways to solve this problem. The method here is just an example.

Since the constraints are small, you can experiment with your code and try different methods.

However, when the constraints are larger, you will have to do some optimisations to lower the runtime of your code.

Common Mistakes

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

int main(){
    int d[6], e, p[7], b[6], a = 0, t;
    bool sp = 0;
    for (int i = 0; i < 6; i++){
        cin >> d[i];
    }
    cin >> e;
    for (int i = 0; i < 7; i++){
        cin >> p[i];
    }
    for (int i = 0; i < 6; i++){
        cin >> t;
        for (int j = 0; j < 6; j++){
            if (t == d[j]){
                d[j] = -1;
                a++;
            }
            if (sp == e){
                sp = 1;
            }
        }
    }
    if (a == 6){
        cout << p[0];
    }else if (a == 5 && sp){
        cout << p[1];
    }else if (a == 5){
        cout << p[2];
    }else if (a == 4 && sp){
        cout << p[3];
    }else if (a == 4) {
        cout << p[4];
    }else if (a == 3 && sp){
        cout << p[5];
    }else{
        cout << p[6];
    }
}
```

Note: The code has been formatted slightly.

Wrong comparison for the special number!

No handling for the no prize special case!

As the code is long with many cases, we might easily miss a mistake like this. Therefore, we should write simpler code.

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

int main(){
    int drawn[6], E, prize[7], ticket[6], on = 0, halfon = 0;
    for (int i = 0; i < 6; i++) {
        cin >> drawn[i];
    }
    cin >> E;
    for (int i = 0; i < 7; i++){
        cin >> prize[i];
    }
    for (int i = 0; i < 6; i++){
        cin >> ticket[i];
    }
    for (int i = 0; i < 6; i++){
        for (int j = 0; j < 6; j++){
            if (ticket[i] == drawn[j]){
                on++;
                break;
            }
        }
        if (ticket[i] = E) {
            halfon = 1;
        }
    }
    if (on == 3){
        cout << prize[6];
    }else if (on == 3 && halfon == 1){
        cout << prize[5];
    }else if (on == 4) {
        cout << prize[4];
    }else if (on == 4 && halfon == 1){
        cout << prize[3];
    }else if (on == 5) {
        cout << prize[2];
    }else if (on == 5 && halfon == 1){
        cout << prize[1];
    }else if (on == 6){
        cout << prize[0];
    }else{
        cout << 0;
    }
}
```

Common Mistakes

Note: The code has been formatted slightly.

Used single = for equality check again!

As the ? matches with special number prize satisfies the criteria for the plain ? matches, it gets overshadowed and every case gets treated like no special number match. We need to fix the order of the cases.

As the code is long with many cases, we might easily miss a mistake like this. Therefore, we should write simpler code.

Candy World

TS269

Problem Statement

Find a subarray with its sum divisible by K.

Fun Fact

There was once a way harder problem in this Team Selection Test, that was proposed by me to prevent people from getting full score!

However, Jack Wong said that there will at most be one 1000/1000.

Therefore, that problem was removed, despite the fact that I thought it was necessary.

Now, after the contest ended, there were 4 1000/1000s, 4 times the number tat Jack Wong thought originally!

This raises the debate: should that extremely hard problem be kept in the first case?

Subtask 1

$N = 1$

Just check if the one and only element is divisible by K!

Simple!

If you can't do it, I don't know what to say.

Subtask 2

$N = 2$

There are three subarrays, the first element, the second element and the whole array.

Just check all of them.

Easy!

O (	1	)	Expected Score	**10**	Cumulative	15

Subtask 3

$K = 2$

In the first two numbers, if one of them is even, then you are good to go.

If not, then the subarray is just the first two elements.

Handle the edge case $N = 1$.

O (	1	)	Expected Score	15	Cumulative	30

Subtask 4

$K = 3$

In the first three numbers, if one of them is divisible by 3, then you are good to go.

If not, then just check if all three numbers modulo 3 are the same.

If yes, then they sum to a multiple of 3.

If not, you can find an adjacent pair of $1\ 2$ or $2\ 1$ modulo 3.

Handle the edge case $N < 3$.

Huh why is this so similar to the last subtask?

O (	1	)	Expected Score	**18**	Cumulative 48

Subtask 5

$C_i = 1$ if i is odd

$C_i = 2$ if i is even

You can notice that there is a cycle of 2, with the remainder of 3 looping.

With some observation, you can notice that when N is big enough (how big though?), there will be a solution.

If $N \equiv 0 \ (mod\ 3)$, just do it like $1\ 2\ 1\ 2\ \ldots\ 1\ 2\ 1\ 2$.

If $N \equiv 1 \ (mod\ 3)$, just do it like $1\ 2\ 1\ 2\ \ldots\ 2\ 1\ 2\ 1$.

If $N \equiv 2 \ (mod\ 3)$, just do it like $2\ 1\ 2\ 1\ \ldots\ 1\ 2\ 1\ 2$.

Output -1 when there is no solution.

O (N)	Expected Score	**13**	Cumulative	61

Subtask 6

$1 \leq N, K, C_i \leq 100$

There exists a solution where $x = 1$.

Loop through all ending days.

Sum up the number of candies you would get from all those days.

If it is divisible by K, it is a solution.

Output 1 and the number of days you stay there (i.e. the ending day).

O (	N^2	)	Expected Score	**32**	Cumulative	93

Subtask 6 Optimisation

You might notice that you can add the number of candies you get in the next day to the total from the previous day to calculate the new total.

Instead of recalculating, you can use the work before and calculate in constant time.

This is somehow a hint to the later subtasks.

Subtask 7

$1 \leq N, K, C_i \leq 100$

Loop through all pairs of starting and ending days.

Sum up the number of candies you would get from all those days.

If it is divisible by K, it is a solution.

Output the starting date and the number of days you stay there.

If you cannot find a solution after looping, output -1.

O (	N^3	)	Expected Score	**112**	Cumulative	135

Weak Test Case?

The code can somehow AC $K = 2$ and $K = 3$!!!

Is it weak test case? Is it our fault?

Remember that you can find a guaranteed solution the first few elements?

The code will only loop $2N$ times for subtask 2 and $3N$ times for subtask 3.

Even when there is no solution, N is guaranteed to be very small and will not TLE.

Therefore, when you suspect us of making a mistake, think twice and make sure you are now the one who is wrong!

Definitely not coming from the guy who always does that.

Subtask 8

$1 \leq N, K, C_i \leq 5000$

If you are smart, you might be wondering where subtask 7 optimisation is.

Here it is!

If you loop the starting days as the other loop, you can use the same optimisation as subtask 6!

Using this optimisation, you can easily AC this subtask building upon the subtask 7 code.

O (	N^2	)	Expected Score	**144**	Cumulative	157

Subtask 8 Alterative Solution

$1 \leq N, K, C_i \leq 5000$

We need a fast way to find the sum from L to R.

How can we do that?

Partial sum!

We can use it to do the subtask 7 solution calculations in constant time, solving subtask 8.

Refer to the 2025/26 C++ Foundation Course Chapter 10 for more details.

O (	N^2	)	Expected Score	**144**	Cumulative	157

Subtask 8 Slight optimisation

You can do a simple check to AC subtask 5 using the subtask 8 solution.

How?

Just check if it is possible.

If not, output -1 and terminate program.

Why does this work?

The reasoning is similar to the “weak test case” one.

It is left as an exercise for the reader.

O (	N^2	)	Expected Score	**157**	Cumulative	157

Subtask 10

$1 \leq K \leq 2 \times 10^5$

Remember that we have to loop through all possible starting days to find out if a ending day (or the other way around) has a solution?

What if we could check that in constant time?

We need a whole new type of optimisation to solve this problem.

Can you guess what it is? Hashing!

Subtask 10

If have remainder R modulo K on day S, and remainder R modulo K again on day E, then days S to E will be a valid solution!

Since we have at most 2×10^5 possible remainders, we can store them in an array!

Now, keep a running sum modulo K of the number of candies, if that remainder matches up with the remainder on another day, we are done.

Otherwise, set the array at the index of the remainder to be the current day.

After looping through all days, if there is no solution, output -1.

O (N)	Expected Score	**207**	Cumulative	235

Full Solution

Remember the time-memory trade-off for Judge Addict?

We can do something similar to solve this problem.

How? C++ map (again)!

By storing the remainder as the key and the day that it happens on as the value, we can achieve the purpose of the array, but in less memory.

We can just check if the key exists in the map or not. If yes, get the value.

Day "0" has a remainder of 0, so make sure to deal with that to find solutions where you start on day 1.

However, the time complexity of the solution will slightly increase, but still less enough for us to AC (and the memory is O(N) instead of O(K))!

O ($NlgN$) Expected Score **250**

Subtask 9

$K \leq N$

This subtask guarantees that $K \leq 2 \times 10^5$ as $\mathrm{N} \leq 2 \times 10^5$.

This subtask is just points for people who can solve subtask 10.

This subtask also serves as a safety net for people who did not handle the case where there is no solution as it can be proven that there always exists a solution when $K \leq N$. The proof is left as exercise for reader.

Takeaway

Time complexity is key in many OI problems. Slow code will TLE.

There are usually subtasks for worse time complexities.

They might lead you on the path to AC!

Try to optimise your code using different methods that you will learn later.

Common Mistakes

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

int32_t main(){
    int n, k;
    cin >> n >> k;
    int arr[n + 12], ps[n + 12];
    ps[2] = 0;
    map<int, int> m;
    for (int i = 2; i <= n + 1; i++){
        cin >> arr[i];
        ps[i + 1] = (ps[i] + arr[i] % k) % k;
        if (m.count(ps[i + 1])) {
            cout << m[ps[i + 1]] << ' ' << i - m[ps[i + 1]] << endl;
            return 0;
        }else{
            m[ps[i + 1]] = i;
        }
    }
    cout << -1 << endl;
    return 0;
}
```

Note: The code has been formatted slightly.

First of all, this student used weird indexing.

While not inherently incorrect, let's help them fix it.

We just need to shift it to make it look normal.

It will make it easier for us to follow their code.

Common Mistakes

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

int32_t main(){
    int n, k;
    cin >> n >> k;
    int arr[n + 12], ps[n + 12];
    ps[0] = 0;
    map<int, int> m;
    for (int i = 1; i <= n; i++){
        cin >> arr[i];
           ps[i] = (ps[i - 1] + arr[i]) % k;
        if (m.count(ps[i])){
            cout << m[ps[i]] + 1 << ' ' << i - m[ps[i]] << endl;
            return 0;
        }else{
            m[ps[i]] = i;
        }
    }
    cout << -1 << endl;
    return 0;
}
```

There, fixed!

However, their code is still wrong! Why?

It turns out they did not check for solutions that start on day 1!

To do that, we can simply set `m[0] = 0` and the case will be accounted for.

In the end, the student found the bug and successfully solved this problem right before the contest ended to clutch out the AK!

Afterword

Overall Reflections

Overall Reflections

The results for the top scores were surprisingly good this year.

However, even with the huge number of easy subtasks, most people still got a not-too-ideal score.

Therefore, water more!

You should not be afraid of subtasks hiding in hard problems!

You should not have the mindset of all or nothing too!

In your OI journey, you will learn and develop your OI senses more.

Good luck and have fun!

Also, please format and indent your code better. Untidy code is hard to debug.