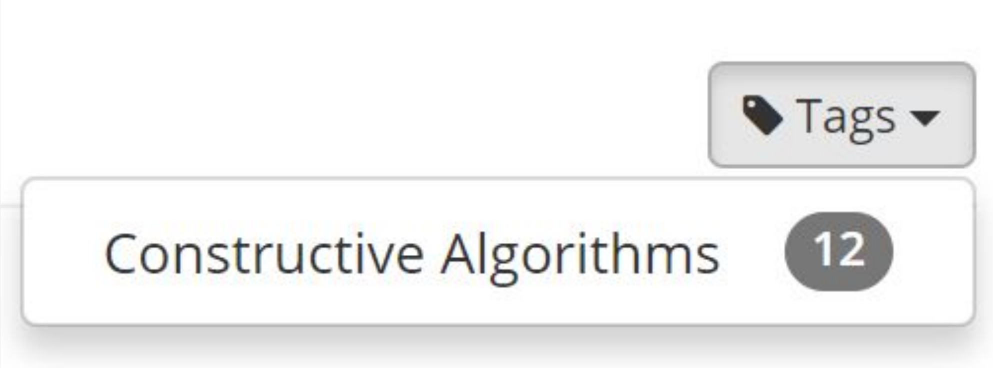
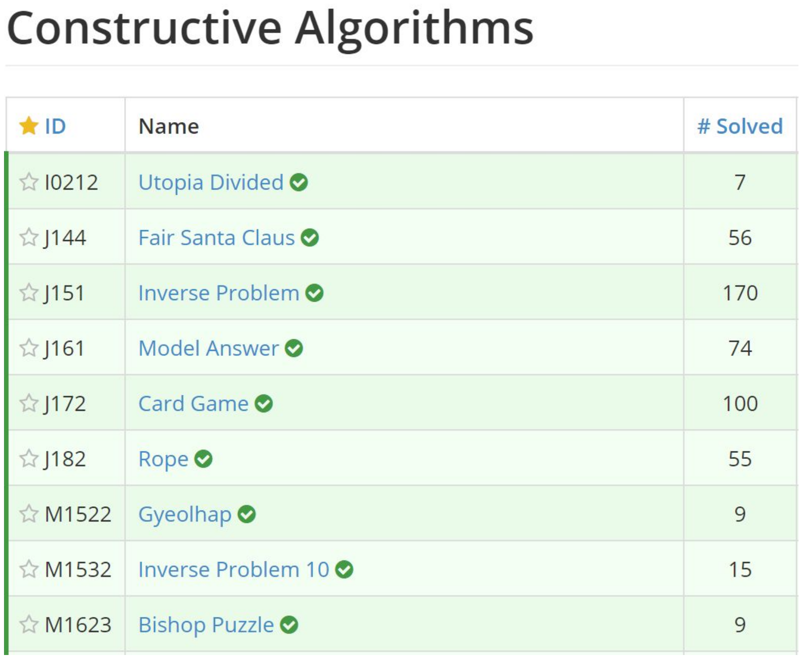
Constructive Algorithms

Percy Wong {percywtc}

What is it?



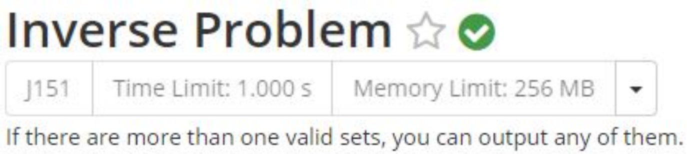
Constructive Algorithms 12



Constructive Algorithms

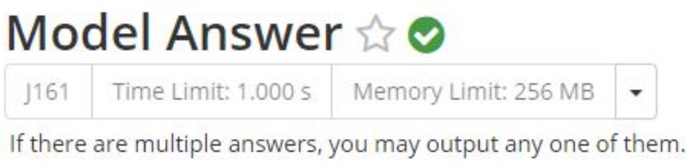
★ ID	Name	# Solved
☆ I0212	Utopia Divided ✓	7
☆ J144	Fair Santa Claus ✓	56
☆ J151	Inverse Problem ✓	170
☆ J161	Model Answer ✓	74
☆ J172	Card Game ✓	100
☆ J182	Rope ✓	55
☆ M1522	Gyeolhap ✓	9
☆ M1532	Inverse Problem 10 ✓	15
☆ M1623	Bishop Puzzle ✓	9

Constructive Problems?


Inverse Problem

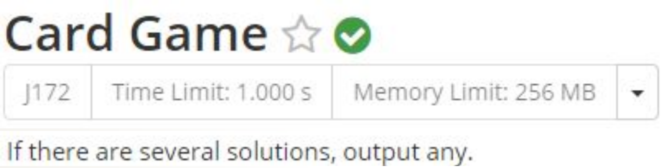
J151 | Time Limit: 1.000 s | Memory Limit: 256 MB

If there are more than one valid sets, you can output any of them.


Model Answer

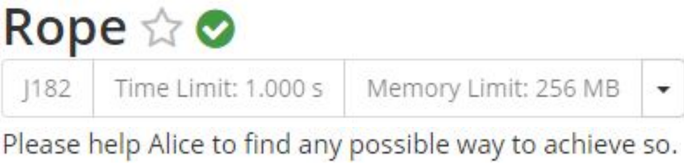
J161 | Time Limit: 1.000 s | Memory Limit: 256 MB

If there are multiple answers, you may output any one of them.


Card Game

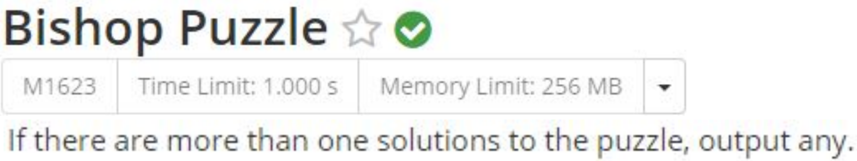
J172 | Time Limit: 1.000 s | Memory Limit: 256 MB

If there are several solutions, output any.


Rope

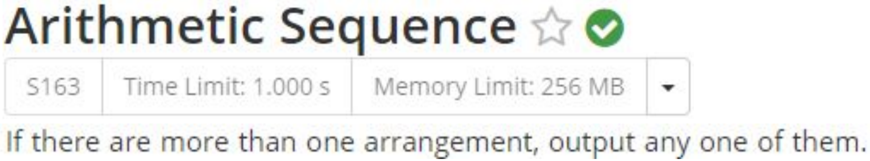
J182 | Time Limit: 1.000 s | Memory Limit: 256 MB

Please help Alice to find any possible way to achieve so.


Bishop Puzzle

M1623 | Time Limit: 1.000 s | Memory Limit: 256 MB

If there are more than one solutions to the puzzle, output any.


Arithmetic Sequence

S163 | Time Limit: 1.000 s | Memory Limit: 256 MB

If there are more than one arrangement, output any one of them.

Constructive Problems...

- Usually give some requirements / constraints to be fulfilled
- You should construct any arrangement that satisfies the given rules
  - Permutations
  - Sequences
  - Matrices
  - Placements
  - ...
- Mostly interesting
- Often require thinking more than coding (standard algorithms)
- May have various correct solutions and “seemingly correct solutions”

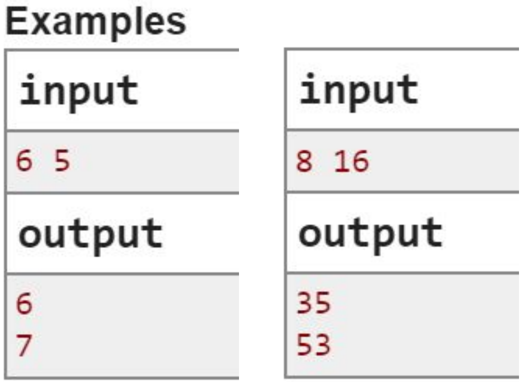
An Example

From Codeforces AIM Tech Round 5 Problem B - Unnatural Conditions

- https://codeforces.com/contest/1028/problem/B

s(x) be sum of digits in decimal representation of positive integer **x**.
Given two integers **n** and **m**, find some positive integers **a** and **b** such that:

- **s(a) ≥ n**
- **s(b) ≥ n**
- **s(a+b) ≤ m**



Examples

input

```
6 5
```

output

```
6
7
```

input

```
8 16
```

output

```
35
53
```

An Example

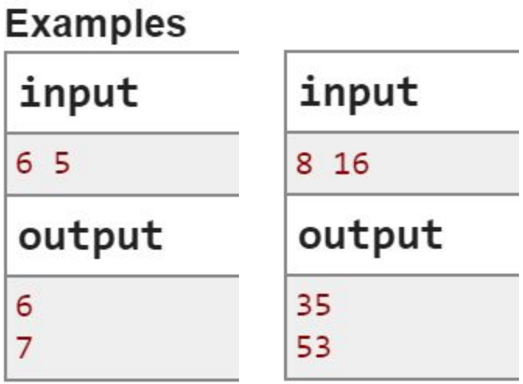
From Codeforces AIM Tech Round 5 Problem B - Unnatural Conditions

- https://codeforces.com/contest/1028/problem/B

s(x) be sum of digits in decimal representation of positive integer **x**
Given two integers **n** and **m**, find some positive integers **a** and **b** such that:

- **s(a) ≥ n**
- **s(b) ≥ n**
- **s(a+b) ≤ m**

1 ≤ n,m ≤ 1129
Both **a** and **b** must have length no more than 2230.



Examples

input

```
6 5
```

output

```
6
7
```

input

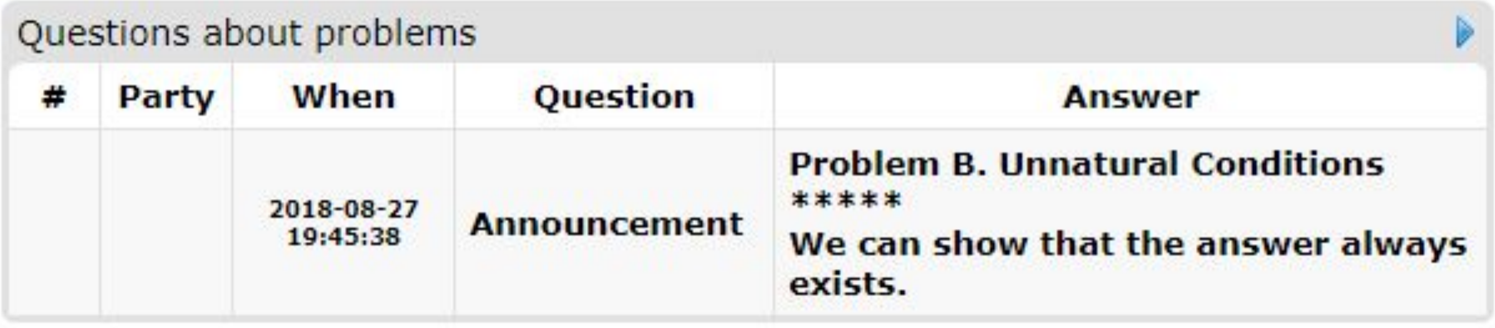
```
8 16
```

output

```
35
53
```

An Example

A big hint to this problem:



Questions about problems

#	Party	When	Question	Answer
		2018-08-27 19:45:38	Announcement	Problem B. Unnatural Conditions ***** We can show that the answer always exists.

An Example

From Codeforces AIM Tech Round 5 Problem B - Unnatural Conditions

- https://codeforces.com/contest/1028/problem/B

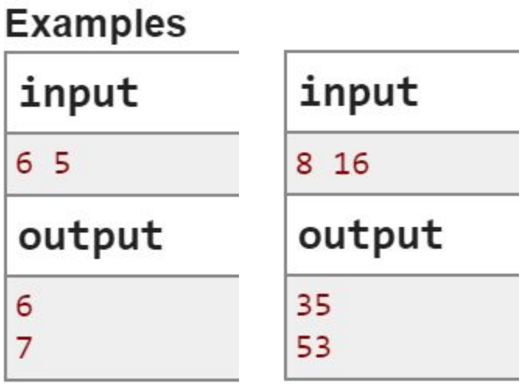
s(x) be sum of digits in decimal representation of positive integer **x**

Given two integers **n** and **m**, find some positive integers **a** and **b** such that:

- **s(a) ≥ n**
- **s(b) ≥ n**
- **s(a+b) ≤ m**

1 ≤ n,m ≤ 1129

Both **a** and **b** must have length no more than 2230.



Examples

input
```
6 5
```
output
```
6
7
```

input
```
8 16
```
output
```
35
53
```

An Example

From Codeforces AIM Tech Round 5 Problem B - Unnatural Conditions

- https://codeforces.com/contest/1028/problem/B

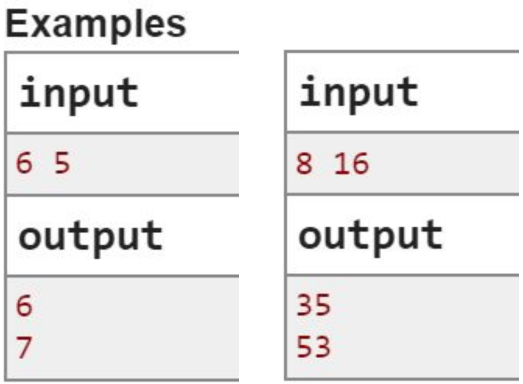
s(x) be sum of digits in decimal representation of positive integer **x**

Given two integers **n** and **m**, find some positive integers **a** and **b** such that:

- **s(a) ≥ *1129***
- **s(b) ≥ 1129**
- **s(a+b) ≤ *1***

1 ≤ n,m ≤ 1129

Both **a** and **b** must have length no more than 2230.



Examples

```
input
6 5
output
6
7
```

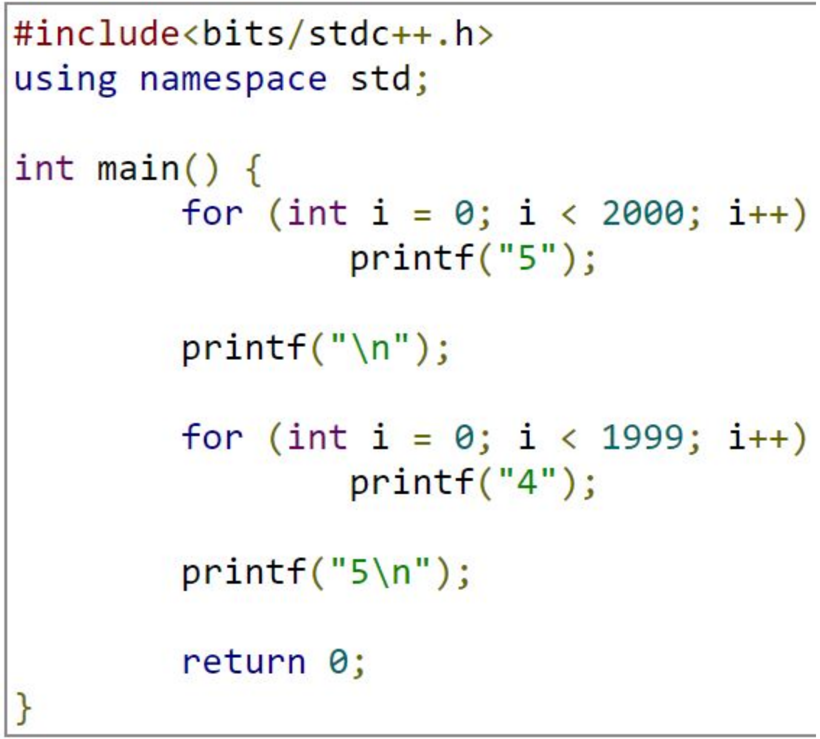
```
input
8 16
output
35
53
```

An Example

a = 55…555 (2000 fives)

b = 44…445 (1999 fours, 1 five)

a+b = 100…000 (1 one, 2000 zeroes)



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

int main() {
        for (int i = 0; i < 2000; i++)
                printf("5");

        printf("\n");

        for (int i = 0; i < 1999; i++)
                printf("4");

        printf("5\n");

        return 0;
}
```

Let’s solve S163 together :)

The problem: given an integer **N**, output a permutation of 1..**N**, such that:

No any triples (**i**, **j**, **k**) where $1 \le \mathbf{i} < \mathbf{j} < \mathbf{k} \le \mathbf{N}$ such that $\mathbf{S}_j - \mathbf{S}_i = \mathbf{S}_k - \mathbf{S}_j$

Examples for **N** = **6**:

Valid: {**3, 5, 4, 1, 2, 6**} and {**5, 1, 6, 3, 2, 4**}

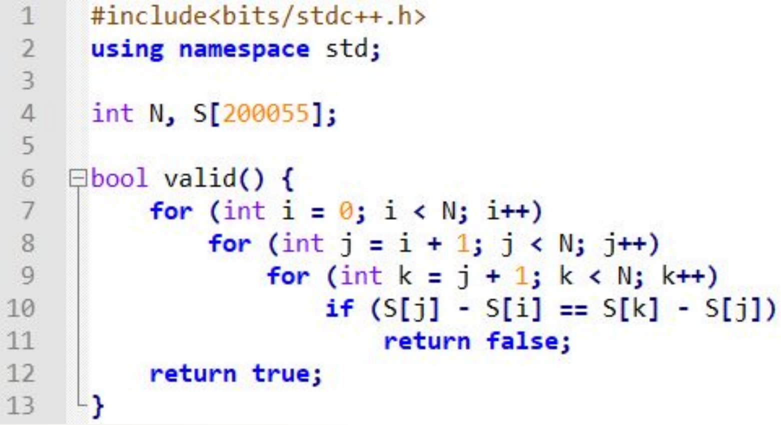
Invalid: {**1**, $\underline{\mathbf{4}}$, **3**, **2**, $\underline{\mathbf{5}}$, $\underline{\mathbf{6}}$} and {$\underline{\mathbf{6}}$, **3**, $\underline{\mathbf{4}}$, **1**, **5**, $\underline{\mathbf{2}}$}

How to approach them?

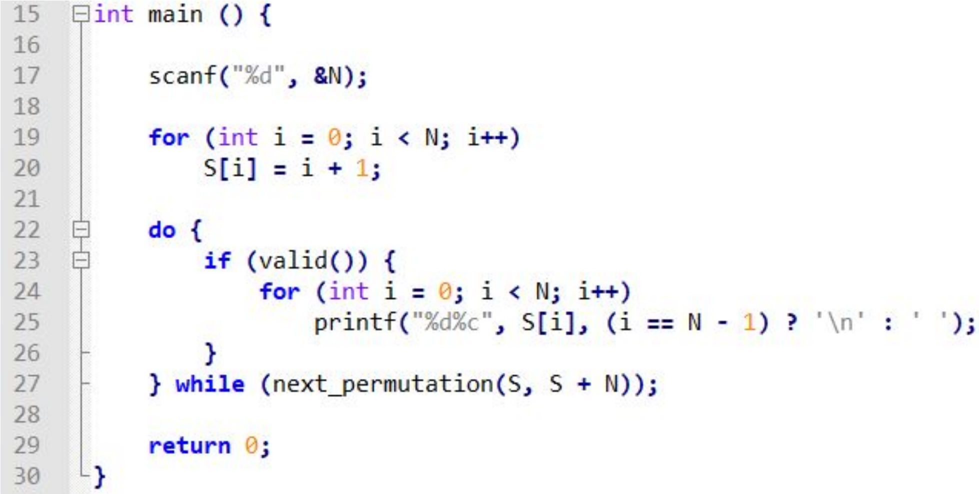
1. Solve some small cases manually / with the aid of programs
2. Observe patterns / relations between them
3. Making some “reasonable” guesses
4. Convince yourself that the guess is correct (or incorrect?)

1. Solving small cases with programs

Write a simple program (not necessarily fast) to exhaust all possibilities



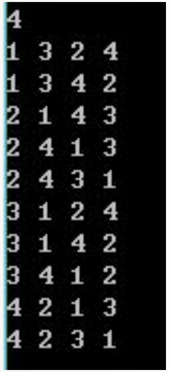
```cpp
#include<bits/stdc++.h>
using namespace std;

int N, S[200055];

bool valid() {
    for (int i = 0; i < N; i++)
        for (int j = i + 1; j < N; j++)
            for (int k = j + 1; k < N; k++)
                if (S[j] - S[i] == S[k] - S[j])
                    return false;
    return true;
}
```



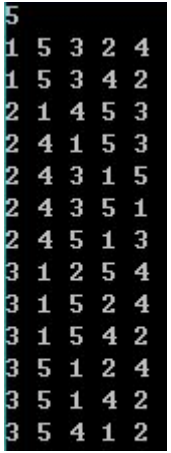
```cpp
int main () {

    scanf("%d", &N);

    for (int i = 0; i < N; i++)
        S[i] = i + 1;

    do {
        if (valid()) {
            for (int i = 0; i < N; i++)
                printf("%d%c", S[i], (i == N - 1) ? '\n' : ' ');
        }
    } while (next_permutation(S, S + N));

    return 0;
}
```

1. Solving small cases with programs

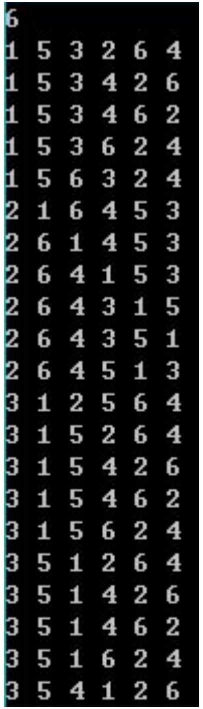
Write a simple program (not necessarily fast) to exhaust all possibilities



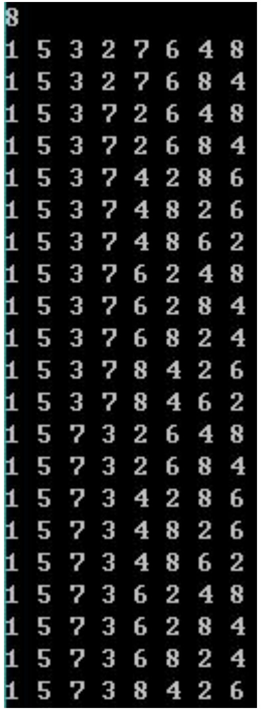
```
4
1 3 2 4
1 3 4 2
2 1 4 3
2 4 1 3
2 4 3 1
3 1 2 4
3 1 4 2
3 4 1 2
4 2 1 3
4 2 3 1
```



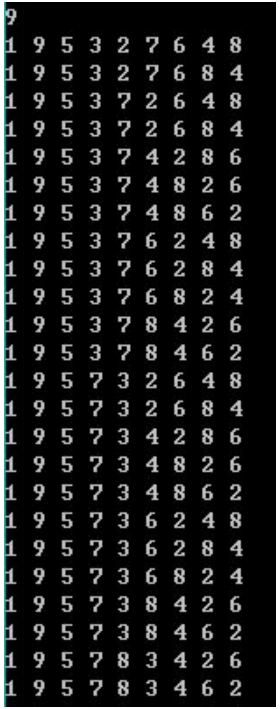
```
5
1 5 3 2 4
1 5 3 4 2
2 1 4 5 3
2 4 1 5 3
2 4 3 1 5
2 4 3 5 1
2 4 5 1 3
3 1 2 5 4
3 1 5 2 4
3 1 5 4 2
3 5 1 2 4
3 5 1 4 2
3 5 4 1 2
```



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



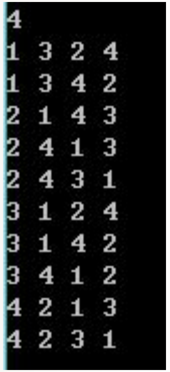
```
8
1 5 3 2 7 6 4 8
1 5 3 2 7 6 8 4
1 5 3 7 2 6 4 8
1 5 3 7 2 6 8 4
1 5 3 7 4 2 8 6
1 5 3 7 4 8 2 6
1 5 3 7 4 8 6 2
1 5 3 7 6 2 4 8
1 5 3 7 6 2 8 4
1 5 3 7 6 8 2 4
1 5 3 7 8 4 2 6
1 5 3 7 8 4 6 2
1 5 7 3 2 6 4 8
1 5 7 3 2 6 8 4
1 5 7 3 4 2 8 6
1 5 7 3 4 8 2 6
1 5 7 3 4 8 6 2
1 5 7 3 6 2 4 8
1 5 7 3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



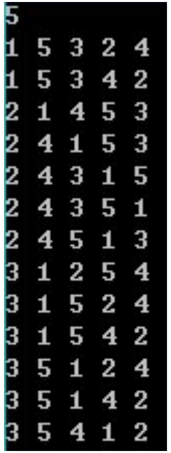
```
9
1 9 5 3 2 7 6 4 8
1 9 5 3 2 7 6 8 4
1 9 5 3 7 2 6 4 8
1 9 5 3 7 2 6 8 4
1 9 5 3 7 4 2 8 6
1 9 5 3 7 4 8 2 6
1 9 5 3 7 4 8 6 2
1 9 5 3 7 6 2 4 8
1 9 5 3 7 6 2 8 4
1 9 5 3 7 6 8 2 4
1 9 5 3 7 8 4 2 6
1 9 5 3 7 8 4 6 2
1 9 5 7 3 2 6 4 8
1 9 5 7 3 2 6 8 4
1 9 5 7 3 4 2 8 6
1 9 5 7 3 4 8 2 6
1 9 5 7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

2. Observe patterns / relations between them

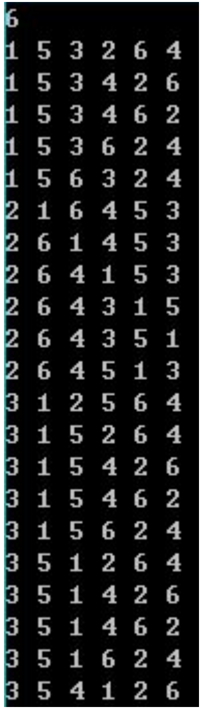
Can you observe any patterns (or phenomena happened on most cases)?



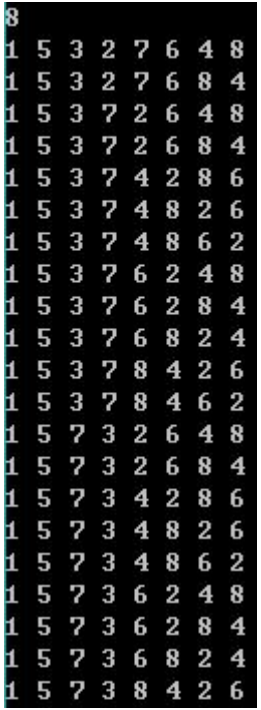
```
4
1 3 2 4
1 3 4 2
2 1 4 3
2 4 1 3
2 4 3 1
3 1 2 4
3 1 4 2
3 4 1 2
4 2 1 3
4 2 3 1
```



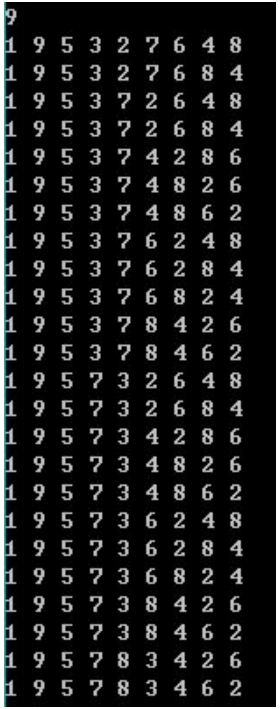
```
5
1 5 3 2 4
1 5 3 4 2
2 1 4 5 3
2 4 1 5 3
2 4 3 1 5
2 4 3 5 1
2 4 5 1 3
3 1 2 5 4
3 1 5 2 4
3 1 5 4 2
3 5 1 2 4
3 5 1 4 2
3 5 4 1 2
```



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



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

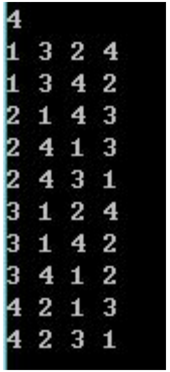


```
9
1 9 5 3 2 7 6 4 8
1 9 5 3 2 7 6 8 4
1 9 5 3 7 2 6 4 8
1 9 5 3 7 2 6 8 4
1 9 5 3 7 4 2 8 6
1 9 5 3 7 4 8 2 6
1 9 5 3 7 4 8 6 2
1 9 5 3 7 6 2 4 8
1 9 5 3 7 6 2 8 4
1 9 5 3 7 6 8 2 4
1 9 5 3 7 8 4 2 6
1 9 5 3 7 8 4 6 2
1 9 5 7 3 2 6 4 8
1 9 5 7 3 2 6 8 4
1 9 5 7 3 4 2 8 6
1 9 5 7 3 4 8 2 6
1 9 5 7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

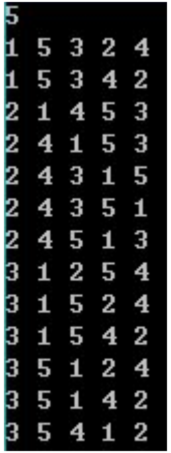
2. Observe patterns / relations between them

Can you observe any patterns (or phenomena happened on most cases)?

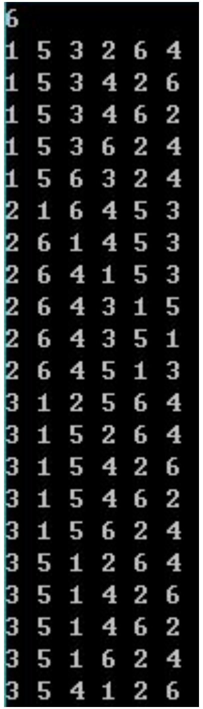
Numbers with the same parity are more likely to be staying together



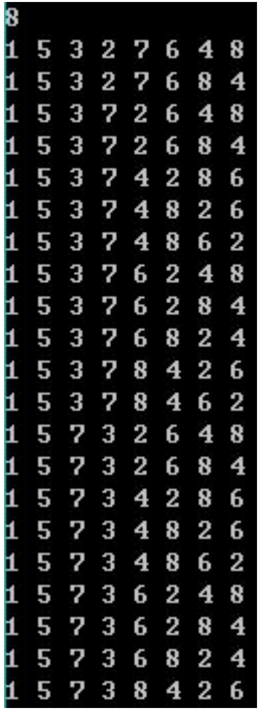
```
4
1  3  2  4
1  3  4  2
2  1  4  3
2  4  1  3
2  4  3  1
3  1  2  4
3  1  4  2
3  4  1  2
4  2  1  3
4  2  3  1
```



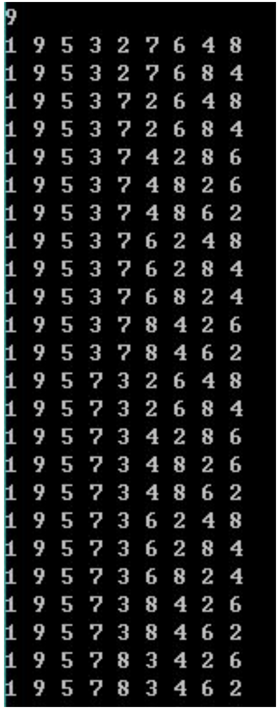
```
5
1  5  3  2  4
1  5  3  4  2
2  1  4  5  3
2  4  1  5  3
2  4  3  1  5
2  4  3  5  1
2  4  5  1  3
3  1  2  5  4
3  1  5  2  4
3  1  5  4  2
3  5  1  2  4
3  5  1  4  2
3  5  4  1  2
```



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



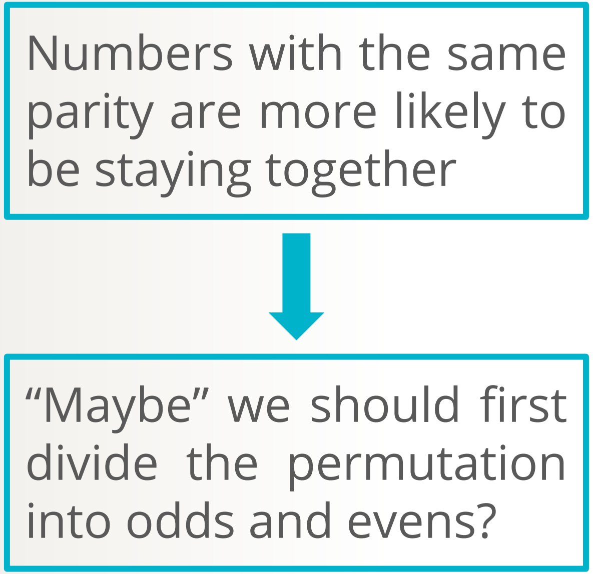
```
8
1  5  3  2  7  6  4  8
1  5  3  2  7  6  8  4
1  5  3  7  2  6  4  8
1  5  3  7  2  6  8  4
1  5  3  7  4  2  8  6
1  5  3  7  4  8  2  6
1  5  3  7  4  8  6  2
1  5  3  7  6  2  4  8
1  5  3  7  6  2  8  4
1  5  3  7  6  8  2  4
1  5  3  7  8  4  2  6
1  5  3  7  8  4  6  2
1  5  7  3  2  6  4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



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

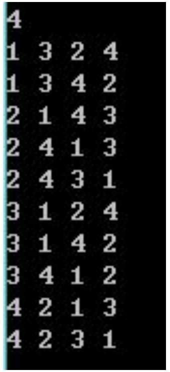
3. Making some “reasonable” guesses

Can you observe any patterns (or phenomena happened on most cases)?

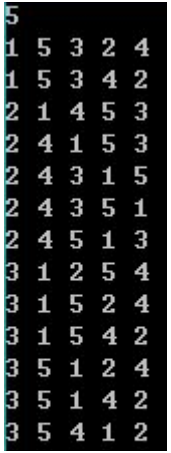


Numbers with the same parity are more likely to be staying together

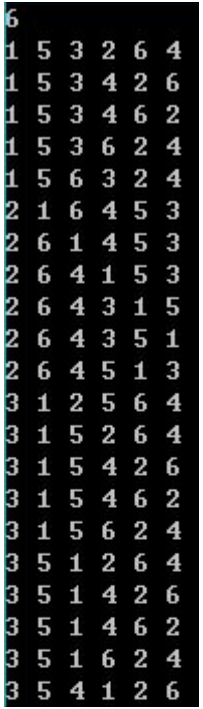
“Maybe” we should first divide the permutation into odds and evens?



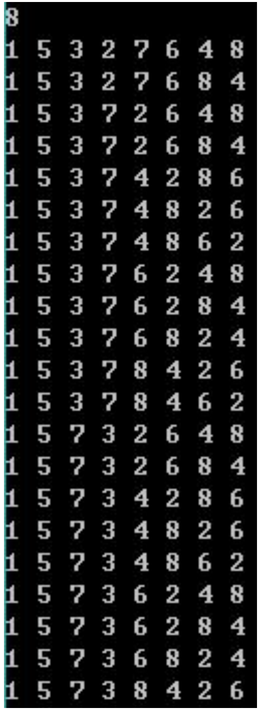
```
4
1 3 2 4
1 3 4 2
2 1 4 3
2 4 1 3
2 4 3 1
3 1 2 4
3 1 4 2
3 4 1 2
4 2 1 3
4 2 3 1
```



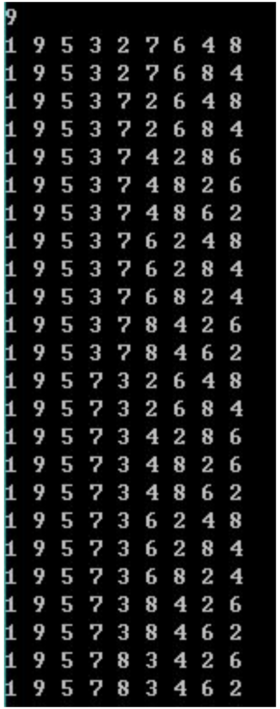
```
5
1 5 3 2 4
1 5 3 4 2
2 1 4 5 3
2 4 1 5 3
2 4 3 1 5
2 4 3 5 1
2 4 5 1 3
3 1 2 5 4
3 1 5 2 4
3 1 5 4 2
3 5 1 2 4
3 5 1 4 2
3 5 4 1 2
```



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



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



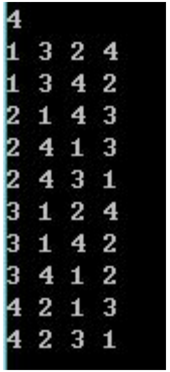
```
9
1 9 5 3 2 7 6 4 8
1 9 5 3 2 7 6 8 4
1 9 5 3 7 2 6 4 8
1 9 5 3 7 2 6 8 4
1 9 5 3 7 4 2 8 6
1 9 5 3 7 4 8 2 6
1 9 5 3 7 4 8 6 2
1 9 5 3 7 6 2 4 8
1 9 5 3 7 6 2 8 4
1 9 5 3 7 6 8 2 4
1 9 5 3 7 8 4 2 6
1 9 5 3 7 8 4 6 2
1 9 5 7 3 2 6 4 8
1 9 5 7 3 2 6 8 4
1 9 5 7 3 4 2 8 6
1 9 5 7 3 4 8 2 6
1 9 5 7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

3. Making some “reasonable” guesses

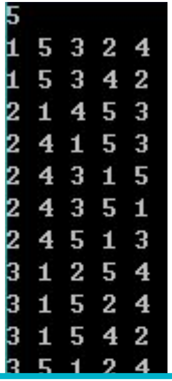
Can you observe any patterns (or phenomena happened on most cases)?

Numbers with the same parity are more likely to be staying together

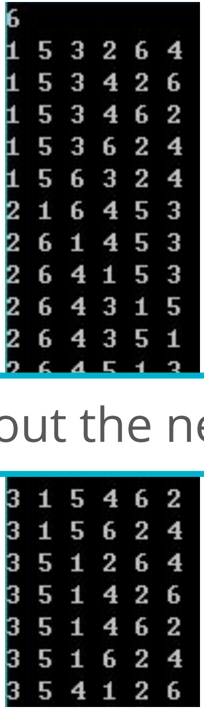
“Maybe” we should first divide the permutation into odds and evens?



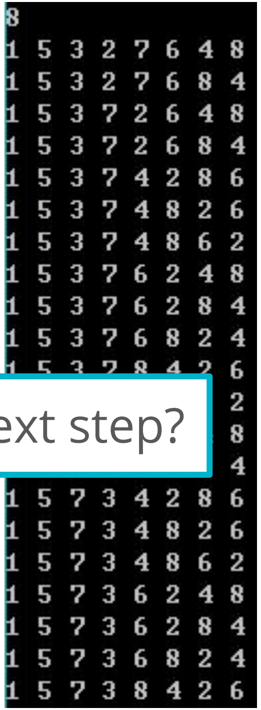
```
4
1 3 2 4
1 3 4 2
2 1 4 3
2 4 1 3
2 4 3 1
3 1 2 4
3 1 4 2
3 4 1 2
4 2 1 3
4 2 3 1
```



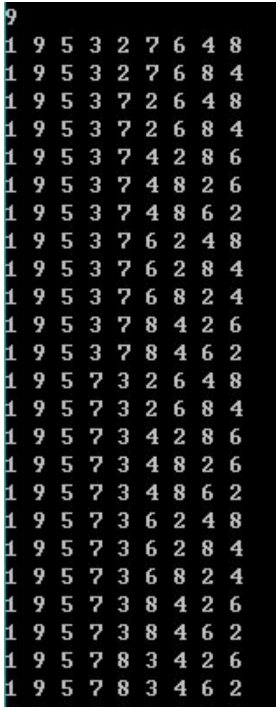
```
5
1 5 3 2 4
1 5 3 4 2
2 1 4 5 3
2 4 1 5 3
2 4 3 1 5
2 4 3 5 1
2 4 5 1 3
3 1 2 5 4
3 1 5 2 4
3 1 5 4 2
3 5 1 2 4
```



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



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



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

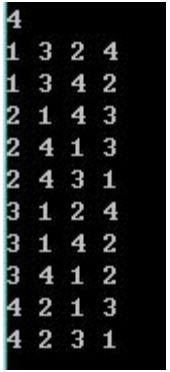
What about the next step?

2. Observe patterns / relations between them

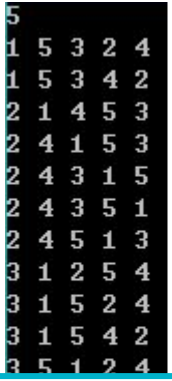
Can you observe any patterns (or phenomena happened on most cases)?

Numbers with the same parity are more likely to be staying together

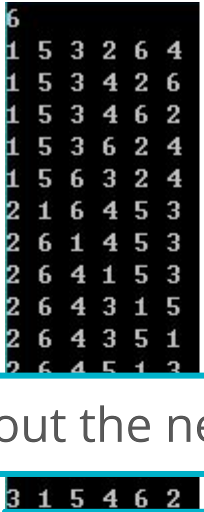
“Maybe” we should first divide the permutation into odds and evens?



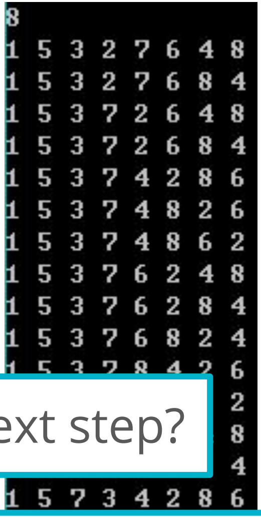
```
4
1 3 2 4
1 3 4 2
2 1 4 3
2 4 1 3
2 4 3 1
3 1 2 4
3 1 4 2
3 4 1 2
4 2 1 3
4 2 3 1
```



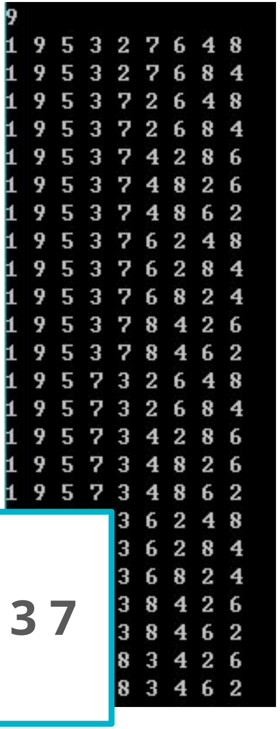
```
5
1 5 3 2 4
1 5 3 4 2
2 1 4 5 3
2 4 1 5 3
2 4 3 1 5
2 4 3 5 1
2 4 5 1 3
3 1 2 5 4
3 1 5 2 4
3 1 5 4 2
3 5 1 2 4
```



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



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



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

What about the next step?

1 5 3 / 1 5 3 7 / 1 9 5 3 7

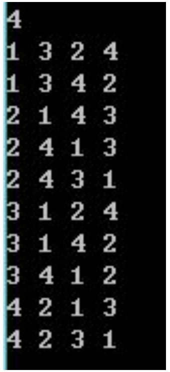
香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

3. Making some “reasonable” guesses

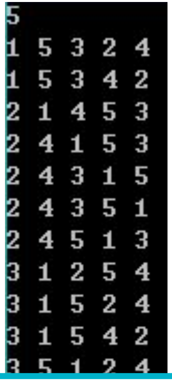
Can you observe any patterns (or phenomena happened on most cases)?

Numbers with the same parity are more likely to be staying together

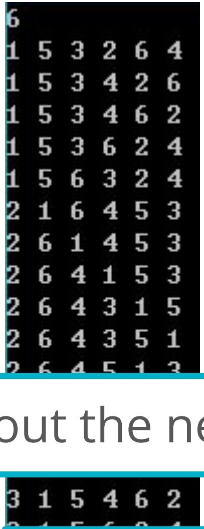
“Maybe” we should first divide the permutation into odds and evens?



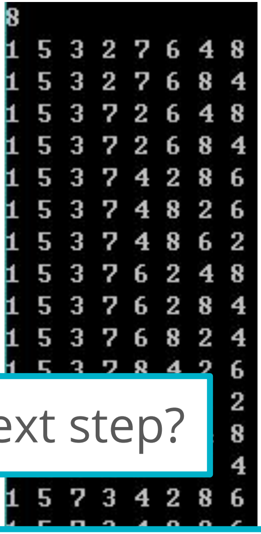
```
4
1 3 2 4
1 3 4 2
2 1 4 3
2 4 1 3
2 4 3 1
3 1 2 4
3 1 4 2
3 4 1 2
4 2 1 3
4 2 3 1
```



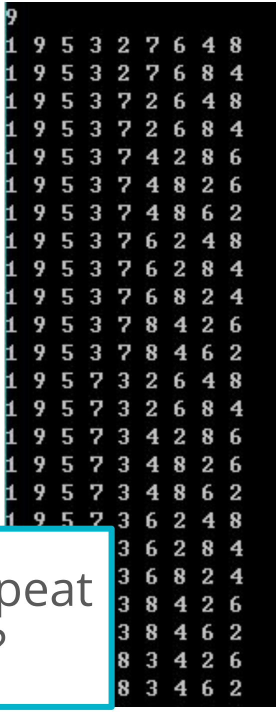
```
5
1 5 3 2 4
1 5 3 4 2
2 1 4 5 3
2 4 1 5 3
2 4 3 1 5
2 4 3 5 1
2 4 5 1 3
3 1 2 5 4
3 1 5 2 4
3 1 5 4 2
3 5 1 2 4
```



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

3 1 5 4 6 2
```



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



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

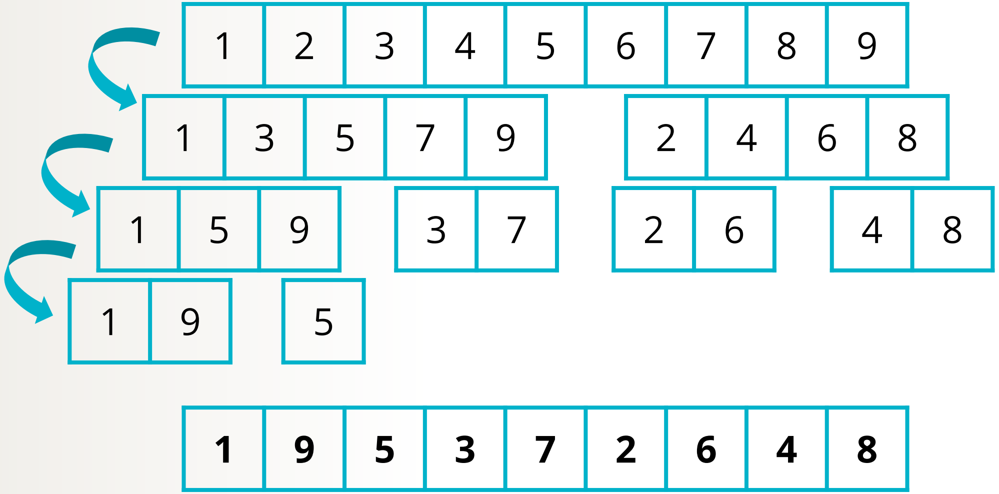
What about the next step?

“Maybe” we can repeat the step recursively?

4. Convince yourself

So we are now trying to use the following solution:

Recursively divide the sequence into two groups by alternatively distributing the numbers to the left and the right



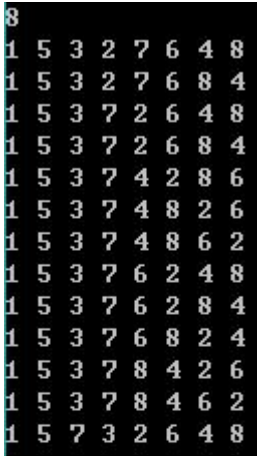
1 2 3 4 5 6 7 8 9

1 3 5 7 9 2 4 6 8

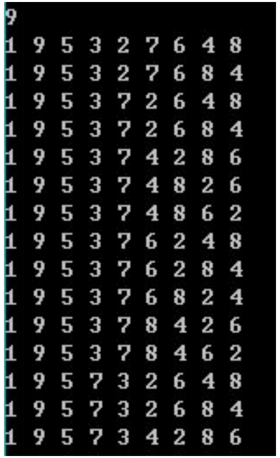
1 5 9 3 7 2 6 4 8

1 9 5

1 9 5 3 7 2 6 4 8

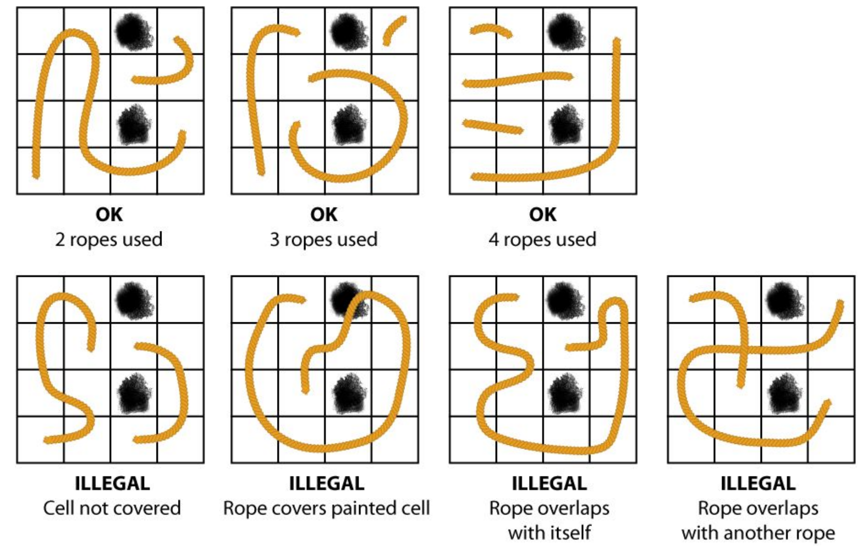


```
8
1 5 3 2 7 6 4 8
1 5 3 2 7 6 8 4
1 5 3 7 2 6 4 8
1 5 3 7 2 6 8 4
1 5 3 7 4 2 8 6
1 5 3 7 4 8 2 6
1 5 3 7 4 8 6 2
1 5 3 7 6 2 4 8
1 5 3 7 6 2 8 4
1 5 3 7 6 8 2 4
1 5 3 7 8 4 2 6
1 5 3 7 8 4 6 2
1 5 7 3 2 6 4 8
```



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

J182 Rope



OK
2 ropes used

OK
3 ropes used

OK
4 ropes used

ILLEGAL
Cell not covered

ILLEGAL
Rope covers painted cell

ILLEGAL
Rope overlaps with itself

ILLEGAL
Rope overlaps with another rope

SCORING

ILLEGAL or **M > R + C + N**	0%
NICE (i.e. N + 1 < M ≤ R + C + N)	60%
EXCELLENT (i.e. M ≤ N + 1)	100%

SUBTASKS

For all cases:

$1 \le R, C \le 300$

$0 \le N < R \times C$

	Points	Constraints
1	8	$R = C = 2$ $N = 0$
2	18	$R = 1$
3	21	$N = 0$
4	53	No additional constraints

J182 Rope

Sometimes it’s hard to tackle the problem by solving small cases with programs...

- It’s hard to code a general exhaustion program
- It’s hard to observe patterns among all possibilities

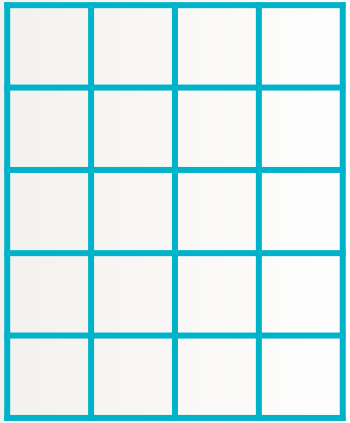
We may try to work on special cases first

J182 Rope

For the entire grid empty (**N** = **0** in Subtask 3),

We can only use **M** = **N** + **1** = **1** rope to fill the whole grid

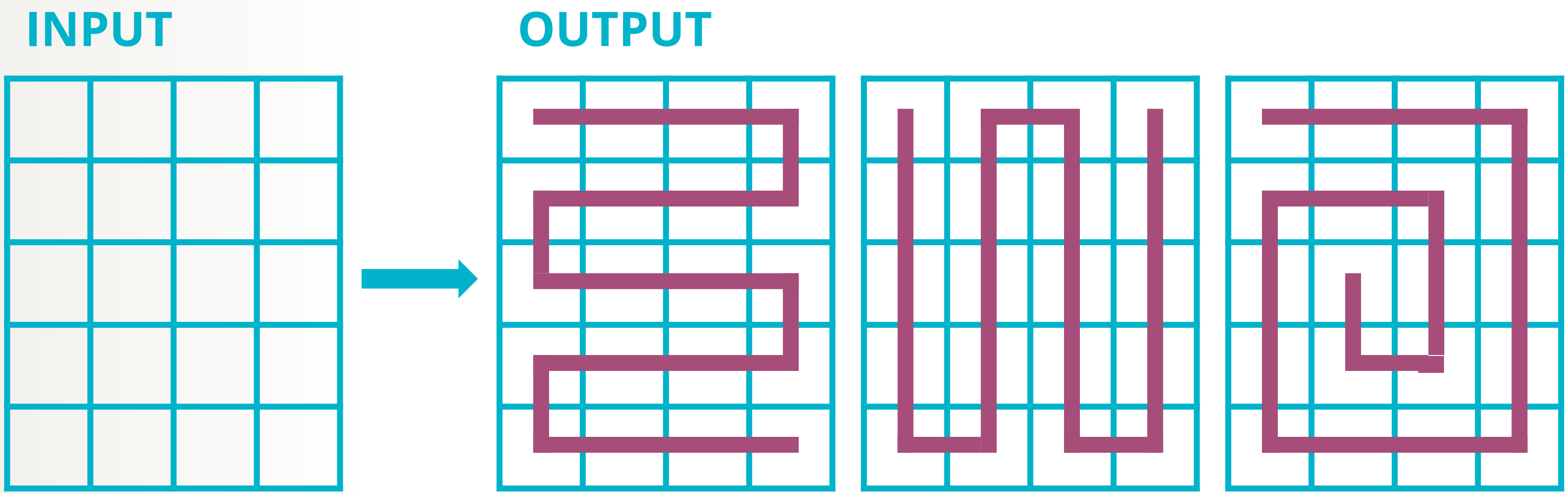
INPUT



J182 Rope

For the entire grid empty (**N = 0** in Subtask 3),

We can only use **M = N + 1 = 1** rope to fill the whole grid



J182 Rope

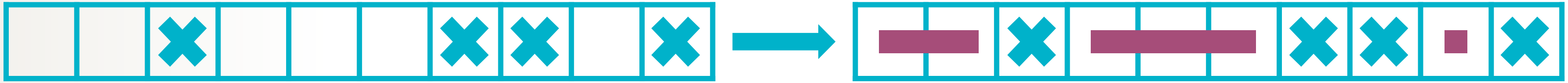
For **R = 1** (Subtask 2),

optimal way is for each consecutive unoccupied interval, place a long rope

It is easy to see that this is the only optimal way

INPUT

OUTPUT



```
1 10 4
1 3
1 7
1 8
1 10
```

```
3
2
1 1
1 2
3
1 4
1 5
1 6
1
1 9
```

J182 Rope

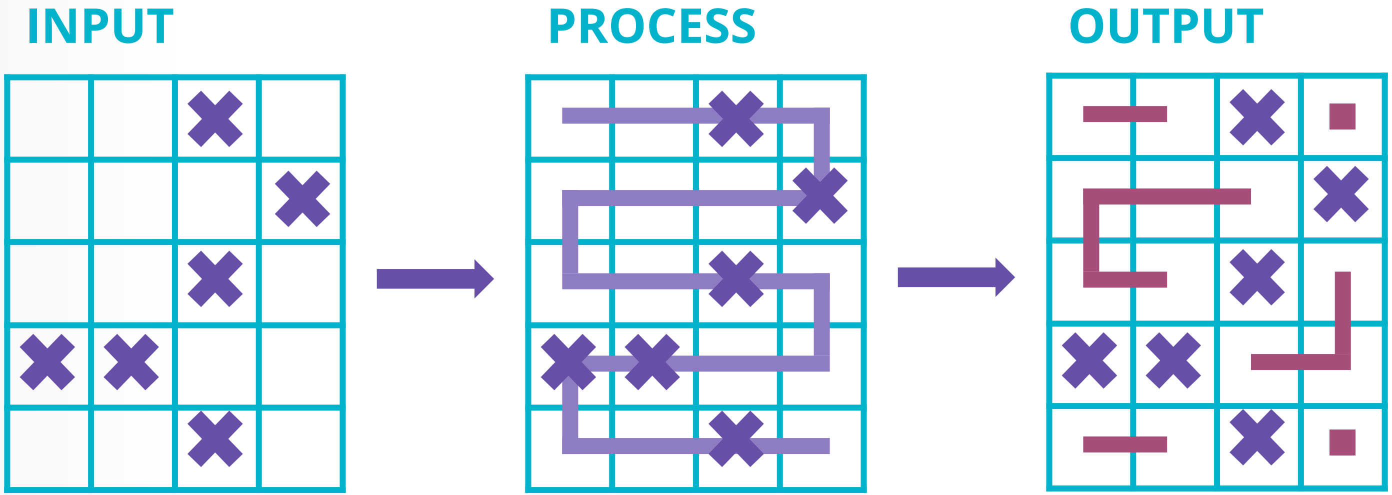
So what about the general case?

Can we obtain general solution from the previous ideas?

J182 Rope - Mix of Previous Ideas

We can consider the **snake** as a **long line**

Using **Long Rope in Interval** on this **long line**



INPUT

PROCESS

OUTPUT

During the break...

Two Problems... :)

- Given array **A** of **N** integers, find any subset that its sum is a multiple of **N**
  - **A** = {**3, 1, 4, 1, 5, 9**}, some solutions (indices in 1-based): {**1, 6**} or {**1, 3, 5**} or {**2, 3, 4**}

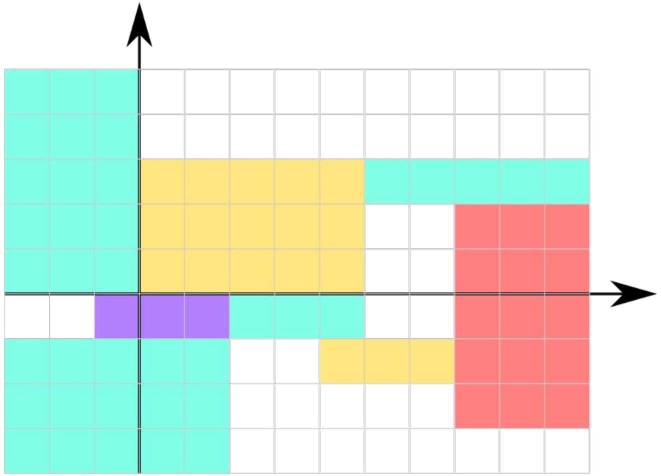
- Given **N**, find **N** consecutive integers that are all composite numbers
  - You may assume that the integers outputted can be infinitely large

Another Problem

Codeforces 763B - Timofey and rectangles

- Input
  - non-overlapping rectangles
  - integer coordinates
  - **odd lengths**

- Output
  - using 4 colors to color the rectangles
  - no touching rectangles share same color

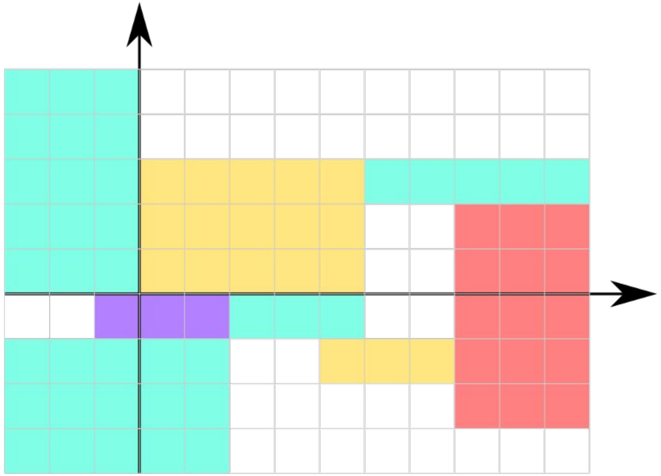


Another Problem

Codeforces 763B - Timofey and rectangles

- Observations
    - Consider the bottom-left cell of rectangles
    - Only different parity in x-coordinate touch (Why?)
    - Same as the y-coordinate

- Idea
    - Coloring according to their parity
    - Combining x- / y-coordinate (How?)



Some more tips...

- Reducing the problem into smaller cases / lower dimensions
- Divide and Conquer
- Greedy Approach
- Binary notations

Some more tips...

- Use some random ideas and carefully analyze why are they incorrect
- Be careful on small / special cases
- Double-check the cases you solve manually / the exhaustion program
- Don’t think too much :)

Some more problems...

- Codeforces
  - Fraction
  - Lesha and array splitting
  - Dasha and Puzzle
  - Puzzling Language (April Fools Contest!!!)
  - Minimum Diameter Tree
  - Seating of Students
  - Construct a tree
- AtCoder
  - Four Coloring

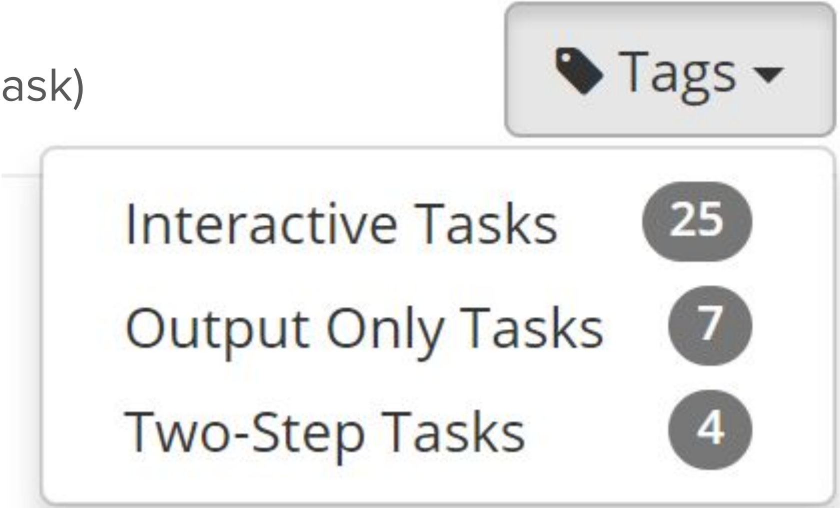
- LS-PC Programming Challenge
  - Annoying Mathematics (2016)
  - Labyrinth (2018)
  - Monorail (2016)
  - Bob the Builder (2018)
  - Gravitational Tetris (2017)
  - Go (2018)

Interactive, Output-only & Communication tasks

Percy Wong {percywtc}

Tasks Categorization

- Batch task
- Interactive task
- Output-only task
- Communication task (a.k.a. Two-steps task)



How Important?

- [IOI2013] Cave [IOI2014] Game [IOI2015] Scales
- [IOI2016] Unscrambling a Messy Bug [IOI2017] The Big Prize
- [IOI2018] Highway Tolls [IOI2018] Combo
- [IOI2010] Maze [IOI2012] Pebbling Odometer [IOI2017] Nowruz
- [IOI2011] Parrots

- [TFT2012] Debug! [TFT2013] The Forgotten Triangle
- [TFT2016] Model Answer II [TFT2018] Cave Exploration
- [TFT2017] Constellation [TFT2018] Exam Anti-Cheat
- [TFT2014] Lost Sequence

- **Interactive**
- **Output-only**
- **Two-steps**

What’s the difficulty?

- Unfamiliar style
- You may not be able to understand these problems during the contests,
  if you are the first time facing new types of tasks

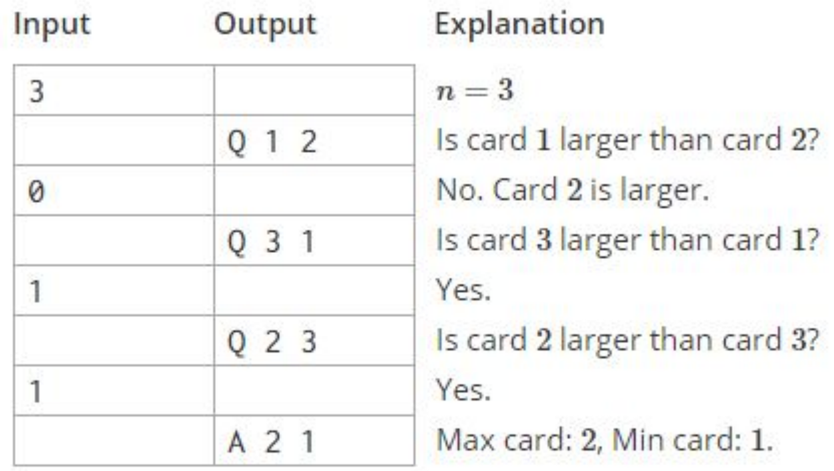
- Feedback from inexperienced contestants after TFTs
  - 「唔知條題目講乜」
  - 「睇唔明題目」
  - 「唔識用 grader」

Interactive task

- Your program will interact with the judging program
- You can consider it as: (suitable for most interactive tasks)
  - Your program asks some questions
  - The judging program answers your questions
  - Repeat the aboves until you can solve “something”
  - (Just like playing MASTERMIND / Guess the Number)
- Usually, there will be limits on number of questions asked
- Or, your score is determined by questions asked

M1431 Comparing Game

- **N** distinct cards not revealed to you
- Your goal: find where are the maximum and the minimum cards
- Question you may ask:
  - “Is card **X** larger than card **Y**?”
- Ask no more than ⌊1.5**N**⌋ questions



Input	Output	Explanation
3		$n = 3$
	Q 1 2	Is card 1 larger than card 2?
0		No. Card 2 is larger.
	Q 3 1	Is card 3 larger than card 1?
1		Yes.
	Q 2 3	Is card 2 larger than card 3?
1		Yes.
	A 2 1	Max card: 2, Min card: 1.

M1431 Comparing Game

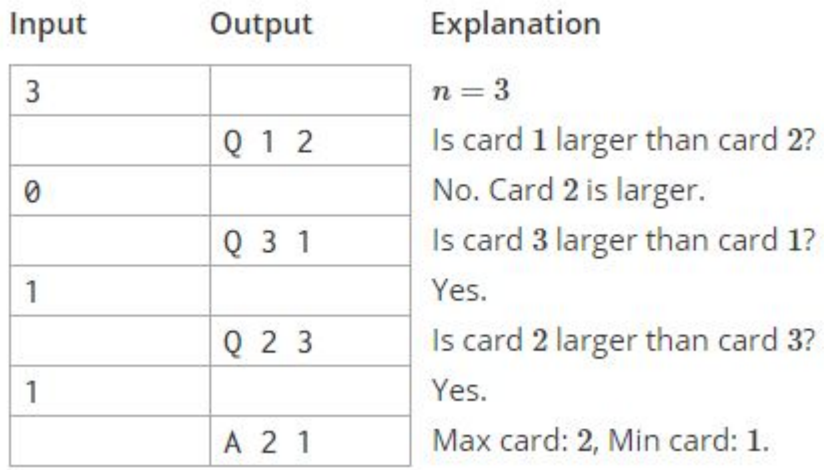
- How can our program asks questions?
  - using standard I/O

Pascal version

```
writeln('Q ', x, ' ', y);
flush(output);    // IMPORTANT
readln(result);
```

C/C++ version

```
printf("Q %d %d\n", x, y);
fflush(stdout);   // IMPORTANT
scanf("%d", &result);
```

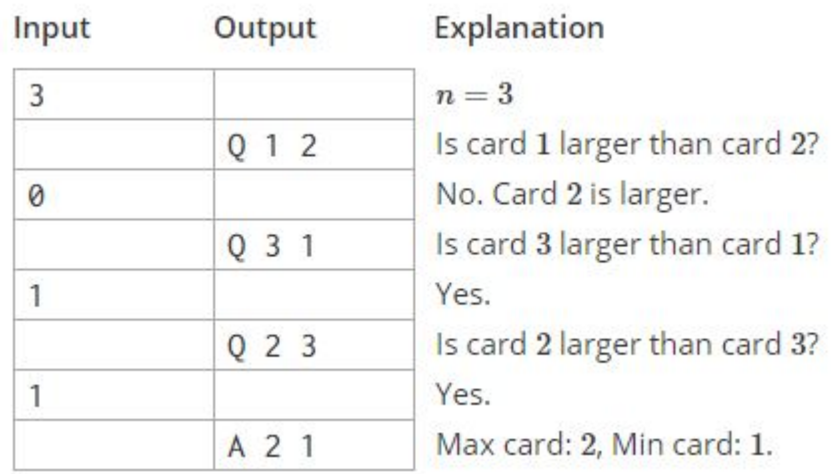


Input	Output	Explanation
3		$n = 3$
	Q 1 2	Is card 1 larger than card 2?
0		No. Card 2 is larger.
	Q 3 1	Is card 3 larger than card 1?
1		Yes.
	Q 2 3	Is card 2 larger than card 3?
1		Yes.
	A 2 1	Max card: 2, Min card: 1.

M1431 Comparing Game

Pascal version (sample partial solution)

```
for i := 1 to N do
  for j := 1 to N do
  begin
    counter := 0;
    if (i <> j) then
    begin
      writeln('Q ', i, ' ', j);
      flush(output);
      readln(result);
      if (result = 1)
        counter := counter + 1;
    end;
    if (counter = N - 1) then
      bigIndex := i;
    if (counter = 0) then
      smallIndex := i;
  end;
```

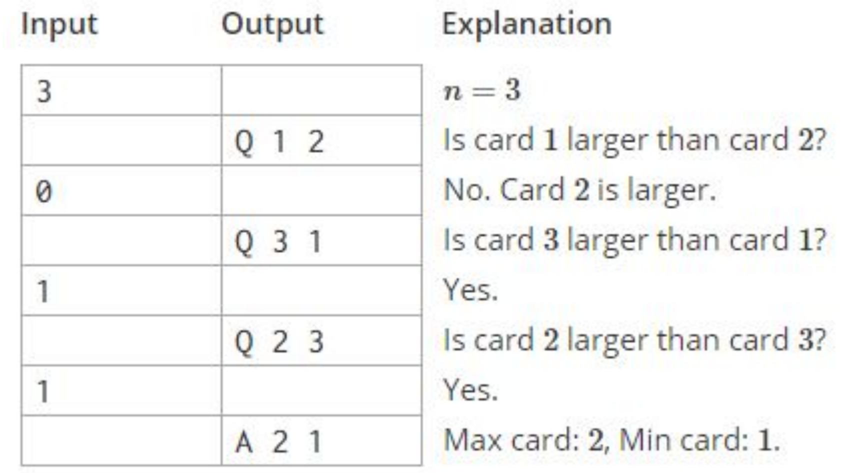


Input	Output	Explanation
3		$n = 3$
	Q 1 2	Is card 1 larger than card 2?
0		No. Card 2 is larger.
	Q 3 1	Is card 3 larger than card 1?
1		Yes.
	Q 2 3	Is card 2 larger than card 3?
1		Yes.
	A 2 1	Max card: 2, Min card: 1.

M1431 Comparing Game

C/C++ version (sample partial solution)

```
for (int i = 1; i <= N; i++)
  for (int j = 1; j <= N; j++) {
    counter = 0;
    if (i != j) {
      printf("Q %d %d\n", i, j);
      fflush(stdout);
      scanf("%d", &result);
      if (result == 1)
        counter++;
    }
    if (counter == N - 1)
      bigIndex = i;
    if (counter == 0)
      smallIndex = i;
  }
```



Input	Output	Explanation
3		$n = 3$
	Q 1 2	Is card 1 larger than card 2?
0		No. Card 2 is larger.
	Q 3 1	Is card 3 larger than card 1?
1		Yes.
	Q 2 3	Is card 2 larger than card 3?
1		Yes.
	A 2 1	Max card: 2, Min card: 1.

M1431 Comparing Game

Recalling that...

- Ask no more than ⌊1.5**N**⌋ questions

We have asked **N(N-1)** questions :(

Some hints to the full solution:

- We can use 0.5**N** questions to split the cards into two groups...
- For **S** numbers, **S-1** comparison is sufficient to find the max/min number...

Interactive task

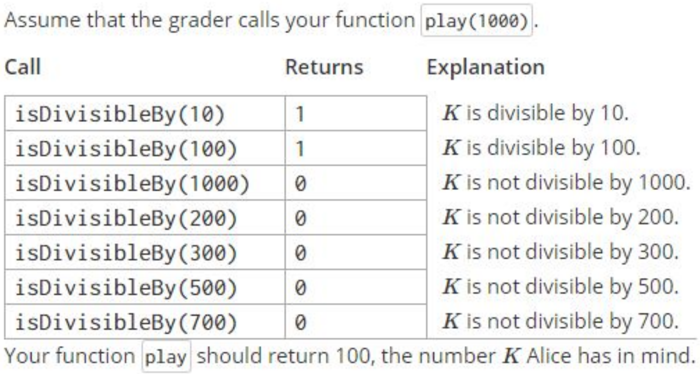
- The example just now performs interaction through standard I/O
  - writeln / printf
  - readln / scanf
- Some interactive tasks are using another way
  - through the grader program
  - you will be given a template code
  - you will ask questions / get feedback by calling some given functions

I0501 Divisor Game

- An unknown integer **K** within the range [1, **N**]
- Your goal: find the value of **K**

- Question you may ask:
  - "Is the number **K** divisible by some integer **x**?"

- Ask minimal questions



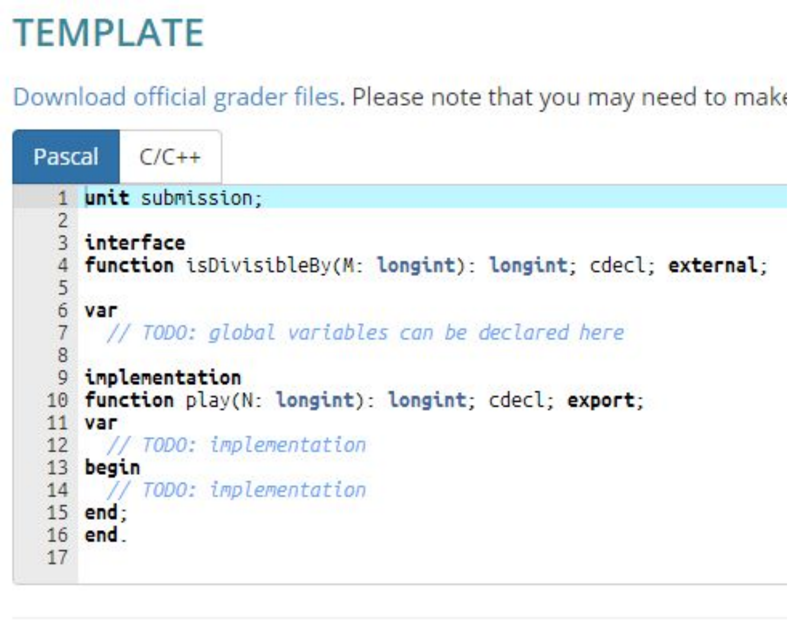
Assume that the grader calls your function `play(1000)`.

Call	Returns	Explanation
`isDivisibleBy(10)`	1	K is divisible by 10.
`isDivisibleBy(100)`	1	K is divisible by 100.
`isDivisibleBy(1000)`	0	K is not divisible by 1000.
`isDivisibleBy(200)`	0	K is not divisible by 200.
`isDivisibleBy(300)`	0	K is not divisible by 300.
`isDivisibleBy(500)`	0	K is not divisible by 500.
`isDivisibleBy(700)`	0	K is not divisible by 700.

Your function `play` should return 100, the number K Alice has in mind.

I0501 Divisor Game

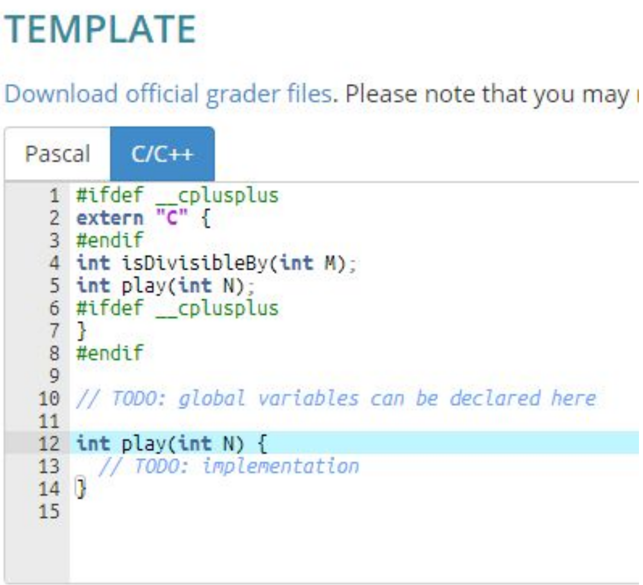
- What is given?



TEMPLATE

Download official grader files. Please note that you may need to make

```pascal
unit submission;

interface
function isDivisibleBy(M: longint): longint; cdecl; external;

var
  // TODO: global variables can be declared here

implementation
function play(N: longint): longint; cdecl; export;
var
  // TODO: implementation
begin
  // TODO: implementation
end;
end.

```



TEMPLATE

Download official grader files. Please note that you may

```c
#ifdef __cplusplus
extern "C" {
#endif
int isDivisibleBy(int M);
int play(int N);
#ifdef __cplusplus
}
#endif

// TODO: global variables can be declared here

int play(int N) {
  // TODO: implementation
}

```

I0501 Divisor Game

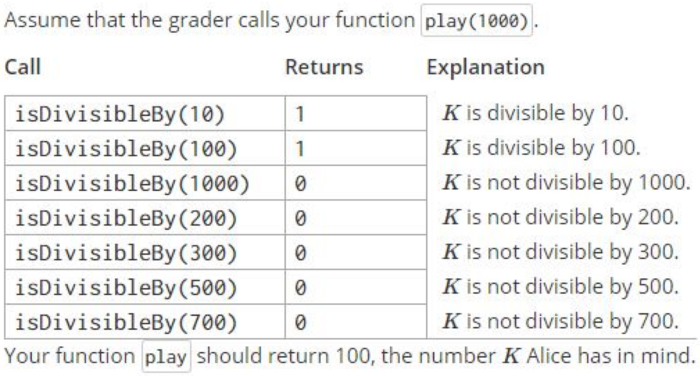
- How can our program ask question?
  - using grader functions

Pascal version

```
result := isDivisibleBy(x);
```

C/C++ version

```
result = isDivisibleBy(x);
```



Assume that the grader calls your function `play(1000)`.

Call	Returns	Explanation
`isDivisibleBy(10)`	1	K is divisible by 10.
`isDivisibleBy(100)`	1	K is divisible by 100.
`isDivisibleBy(1000)`	0	K is not divisible by 1000.
`isDivisibleBy(200)`	0	K is not divisible by 200.
`isDivisibleBy(300)`	0	K is not divisible by 300.
`isDivisibleBy(500)`	0	K is not divisible by 500.
`isDivisibleBy(700)`	0	K is not divisible by 700.

Your function `play` should return 100, the number K Alice has in mind.

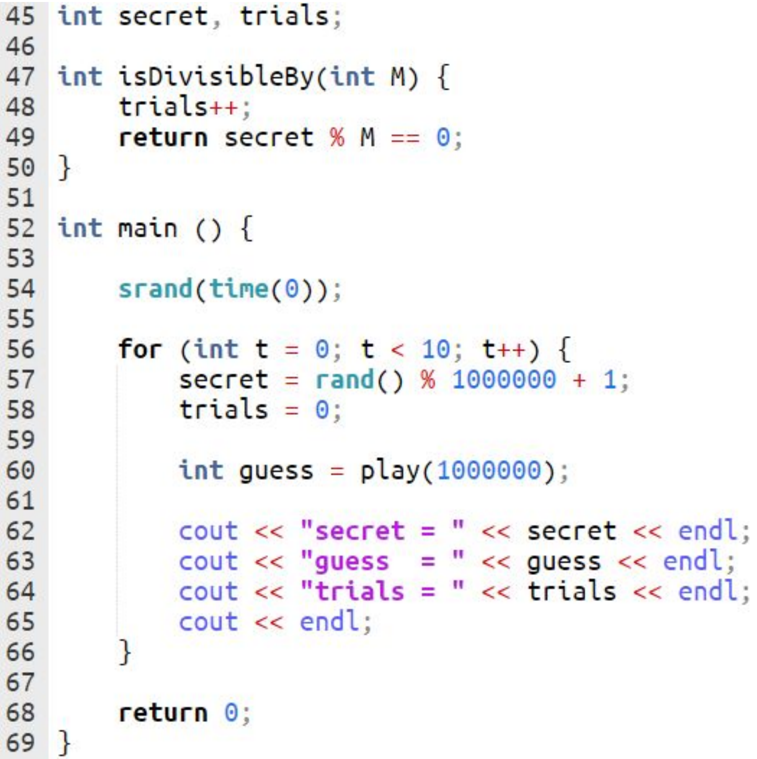
I0501 Divisor Game

- You cannot compile the program even if you have completed `play()`
  - it’s because the main program is missing
- You cannot test the program
  - it’s because the function `isDivisibleBy()` is not implemented
  - this function is implemented by the judging program
  - you are only required to implement `play()`
- So what can we do to test our program?

I0501 Divisor Game

- So what can we do to test our program?
  - we can implement the remaining functions
    - `int isDivisibleBy(int M)`
    - `int main()`

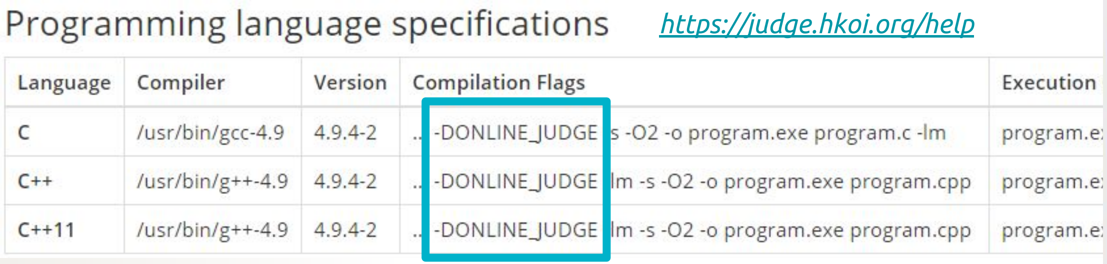
- Delete these parts before submitting
- Or you can use
  - `#ifndef ONLINE_JUDGE`
  - `#endif`



```
int secret, trials;

int isDivisibleBy(int M) {
    trials++;
    return secret % M == 0;
}

int main () {

    srand(time(0));

    for (int t = 0; t < 10; t++) {
        secret = rand() % 1000000 + 1;
        trials = 0;

        int guess = play(1000000);

        cout << "secret = " << secret << endl;
        cout << "guess  = " << guess << endl;
        cout << "trials = " << trials << endl;
        cout << endl;
    }

    return 0;
}
```

I0501 Divisor Game

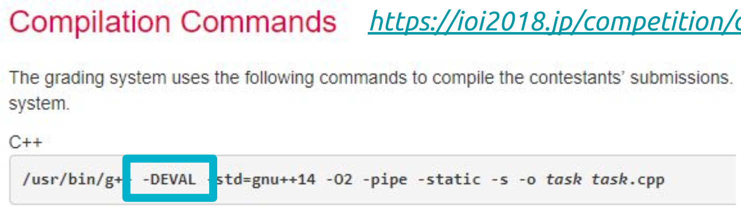
- ~~Delete these parts before submitting~~
- You can use
    - `#ifndef some_flag_here`
    - `#endif`



Programming language specifications

https://judge.hkoi.org/help

Language	Compiler	Version	Compilation Flags	Execution
C	/usr/bin/gcc-4.9	4.9.4-2	.. -DONLINE_JUDGE s -O2 -o program.exe program.c -lm	program.e
C++	/usr/bin/g++-4.9	4.9.4-2	.. -DONLINE_JUDGE lm -s -O2 -o program.exe program.cpp	program.e
C++11	/usr/bin/g++-4.9	4.9.4-2	.. -DONLINE_JUDGE lm -s -O2 -o program.exe program.cpp	program.e



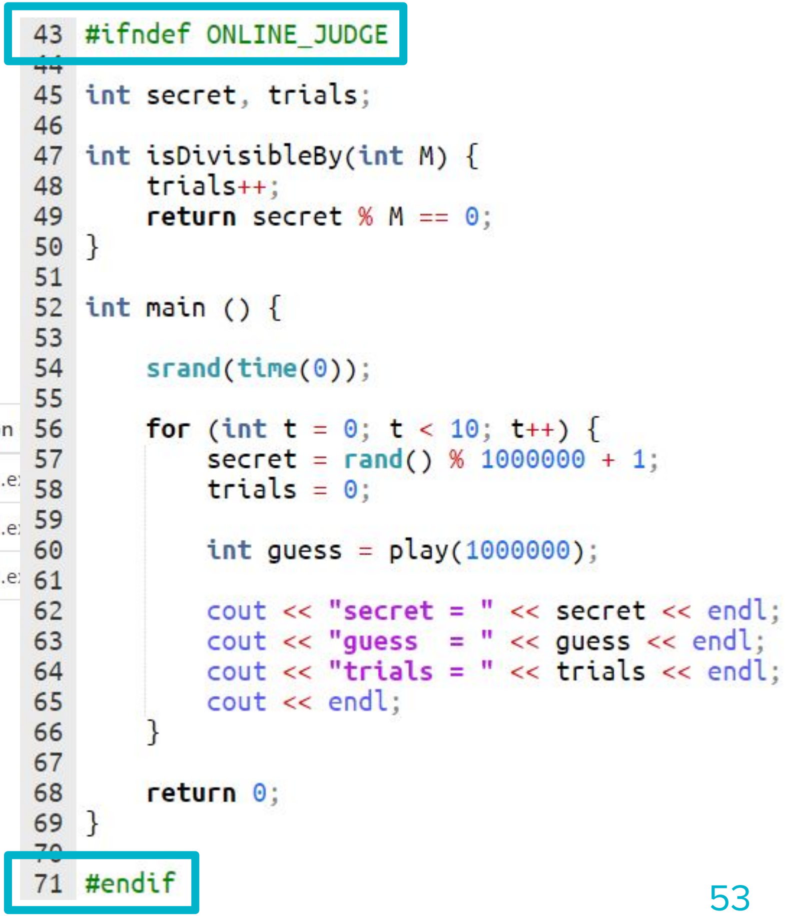
Compilation Commands

https://ioi2018.jp/competition/competition-environment/

The grading system uses the following commands to compile the contestants' submissions. system.

C++

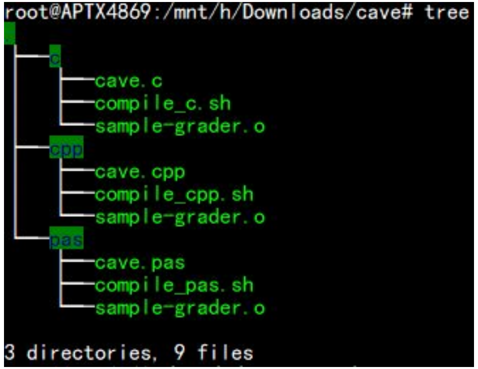
```
/usr/bin/g+ -DEVAL std=gnu++14 -O2 -pipe -static -s -o task task.cpp
```



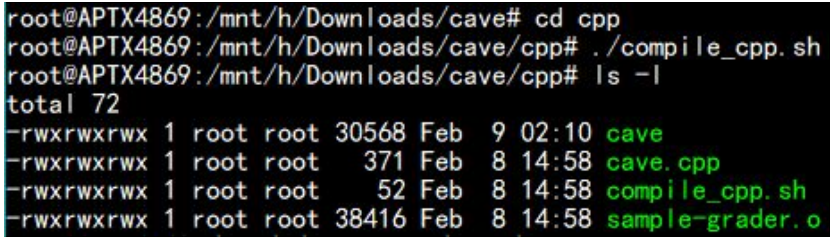
```
#ifndef ONLINE_JUDGE

int secret, trials;

int isDivisibleBy(int M) {
    trials++;
    return secret % M == 0;
}

int main () {

    srand(time(0));

    for (int t = 0; t < 10; t++) {
        secret = rand() % 1000000 + 1;
        trials = 0;

        int guess = play(1000000);

        cout << "secret = " << secret << endl;
        cout << "guess  = " << guess << endl;
        cout << "trials = " << trials << endl;
        cout << endl;
    }

    return 0;
}

#endif
```

T182 Cave Exploration

- Some problems (like T182) provide sample grader files for your testing
- So you don’t need to implement other functions by yourselves… hurray!!?
    - Make sure that you know how to use them :(

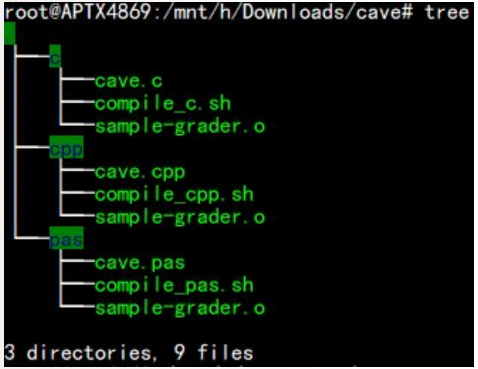


```
root@APTX4869:/mnt/h/Downloads/cave# tree
.
├── c
│   ├── cave.c
│   ├── compile_c.sh
│   └── sample-grader.o
├── cpp
│   ├── cave.cpp
│   ├── compile_cpp.sh
│   └── sample-grader.o
└── pas
    ├── cave.pas
    ├── compile_pas.sh
    └── sample-grader.o

3 directories, 9 files
```

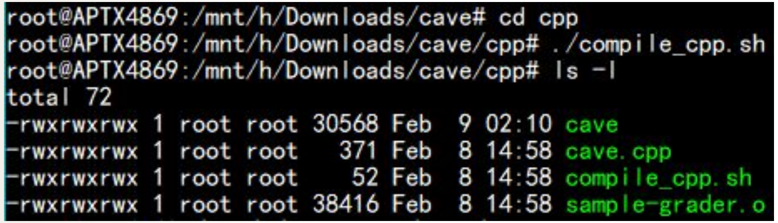


```
root@APTX4869:/mnt/h/Downloads/cave# cd cpp
root@APTX4869:/mnt/h/Downloads/cave/cpp# ./compile_cpp.sh
root@APTX4869:/mnt/h/Downloads/cave/cpp# ls -l
total 72
-rwxrwxrwx 1 root root 30568 Feb  9 02:10 cave
-rwxrwxrwx 1 root root   371 Feb  8 14:58 cave.cpp
-rwxrwxrwx 1 root root    52 Feb  8 14:58 compile_cpp.sh
-rwxrwxrwx 1 root root 38416 Feb  8 14:58 sample-grader.o
```

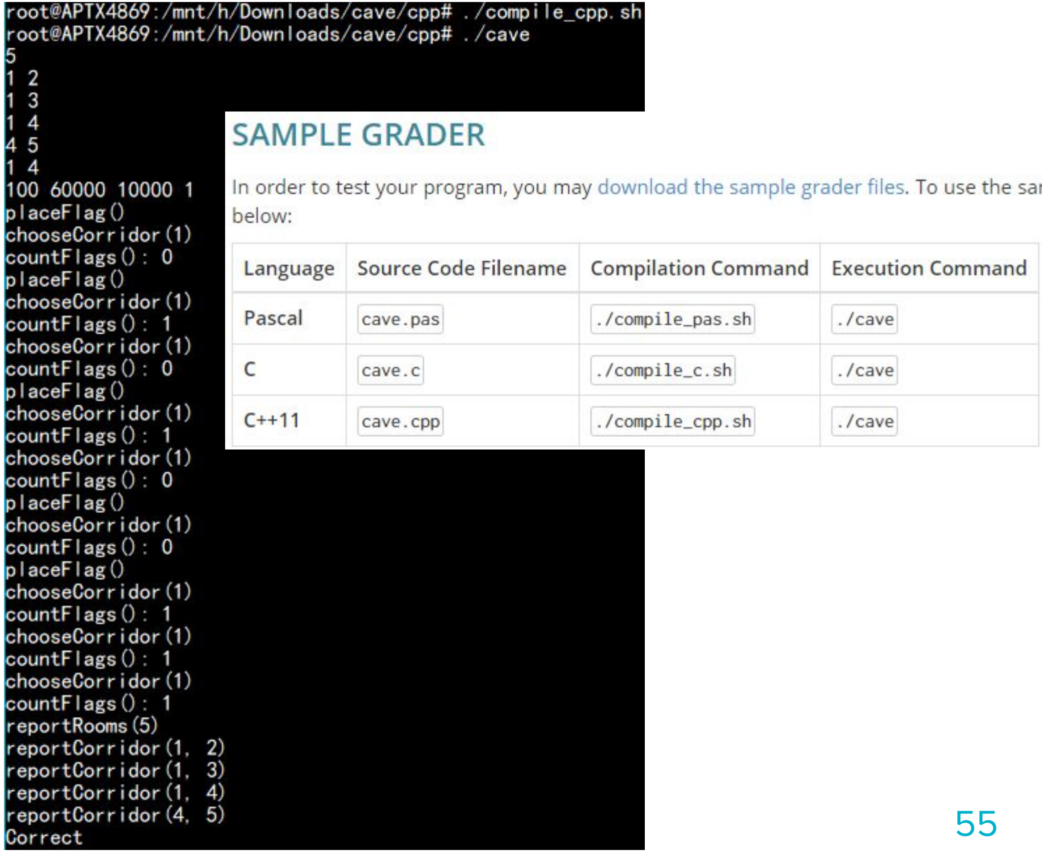
T182 Cave Exploration



```
root@APTX4869:/mnt/h/Downloads/cave# tree
.
├── c
│   ├── cave.c
│   ├── compile_c.sh
│   └── sample-grader.o
├── cpp
│   ├── cave.cpp
│   ├── compile_cpp.sh
│   └── sample-grader.o
└── pas
    ├── cave.pas
    ├── compile_pas.sh
    └── sample-grader.o

3 directories, 9 files
```



```
root@APTX4869:/mnt/h/Downloads/cave# cd cpp
root@APTX4869:/mnt/h/Downloads/cave/cpp# ./compile_cpp.sh
root@APTX4869:/mnt/h/Downloads/cave/cpp# ls -l
total 72
-rwxrwxrwx 1 root root 30568 Feb  9 02:10 cave
-rwxrwxrwx 1 root root   371 Feb  8 14:58 cave.cpp
-rwxrwxrwx 1 root root    52 Feb  8 14:58 compile_cpp.sh
-rwxrwxrwx 1 root root 38416 Feb  8 14:58 sample-grader.o
```



```
root@APTX4869:/mnt/h/Downloads/cave/cpp# ./compile_cpp.sh
root@APTX4869:/mnt/h/Downloads/cave/cpp# ./cave
5
1 2
1 3
1 4
4 5
1 4
100 60000 10000 1
placeFlag()
chooseCorridor(1)
countFlags(): 0
placeFlag()
chooseCorridor(1)
countFlags(): 1
chooseCorridor(1)
countFlags(): 0
placeFlag()
chooseCorridor(1)
countFlags(): 1
chooseCorridor(1)
countFlags(): 0
placeFlag()
chooseCorridor(1)
countFlags(): 0
placeFlag()
chooseCorridor(1)
countFlags(): 1
chooseCorridor(1)
countFlags(): 1
chooseCorridor(1)
countFlags(): 1
reportRooms(5)
reportCorridor(1, 2)
reportCorridor(1, 3)
reportCorridor(1, 4)
reportCorridor(4, 5)
Correct
```

SAMPLE GRADER

In order to test your program, you may download the sample grader files. To use the sa
below:

Language	Source Code Filename	Compilation Command	Execution Command
Pascal	cave.pas	./compile_pas.sh	./cave
C	cave.c	./compile_c.sh	./cave
C++11	cave.cpp	./compile_cpp.sh	./cave

Some more...

- Interactive Problems: Guide for Participants from Codeforces

Practice problems:

- 01084 Celebrity from HKOI Online Judge
- I1021 Memory available on HKOI Online Judge
- T054 Guess from HKOI Online Judge
- Go, Gopher! from Google Code Jam 2018 Qualification Round
- some other problems... suggested by the Codeforecs community

Output-only task

- Formal Definition
    - Input files are given to you
    - You are not required to upload any source codes, just the output files

- Actual meaning
    - No need to worry about failing some unknown cases, all cases are revealed :D
    - No time limits / memory limits (actually there are... TL = 5hrs, ML = your machine)
    - You can even solve the cases manually :D :D :D

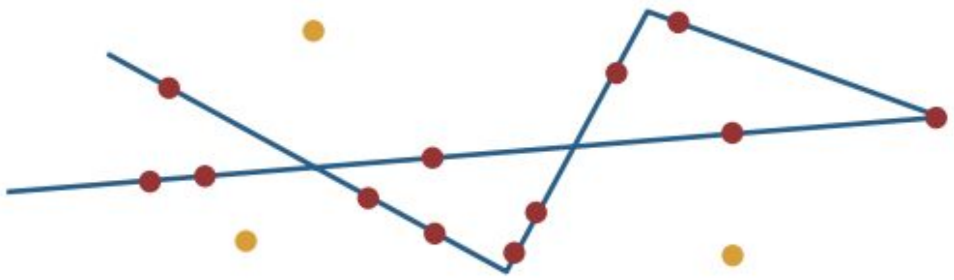
Output-only task

- Common stuffs?
  - not expecting optimal solutions (or not even exist)
  - some formulas to determine how good your outputs are (and how much you score)
  - good-enough solutions can get good-enough scores

- What you can do?
  - Usually there exists some small cases (can be manually solved)
  - You can write programs to analyze the cases / solve the cases
  - You can even solve cases separately with different approach and codes

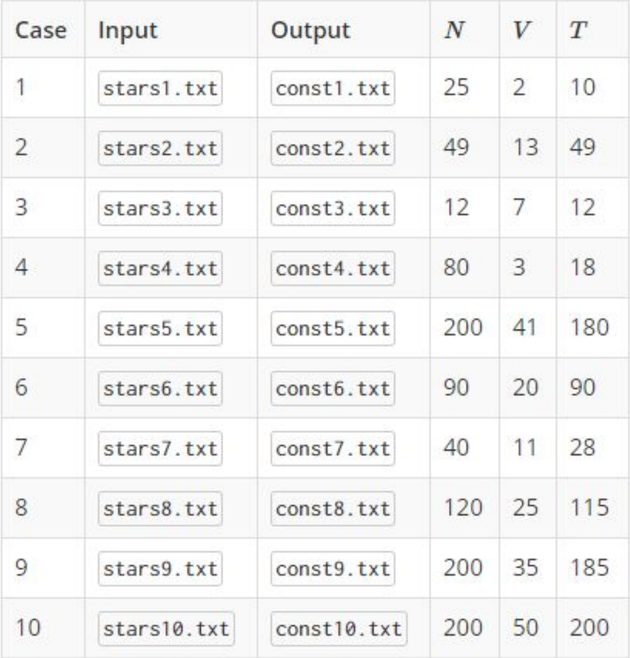
T174 Constellation

- Given a set of **N** points with integral coordinates lying on xy-plane
- Build a polyline consisting of **V** points, connecting most points



T174 Constellation

- 10 cases in total (each 10pts)
- Scoring are based on number of points you connect
    - $10 \times 10^{(P - V) / (T - V)}$



Case	Input	Output	N	V	T
1	`stars1.txt`	`const1.txt`	25	2	10
2	`stars2.txt`	`const2.txt`	49	13	49
3	`stars3.txt`	`const3.txt`	12	7	12
4	`stars4.txt`	`const4.txt`	80	3	18
5	`stars5.txt`	`const5.txt`	200	41	180
6	`stars6.txt`	`const6.txt`	90	20	90
7	`stars7.txt`	`const7.txt`	40	11	28
8	`stars8.txt`	`const8.txt`	120	25	115
9	`stars9.txt`	`const9.txt`	200	35	185
10	`stars10.txt`	`const10.txt`	200	50	200

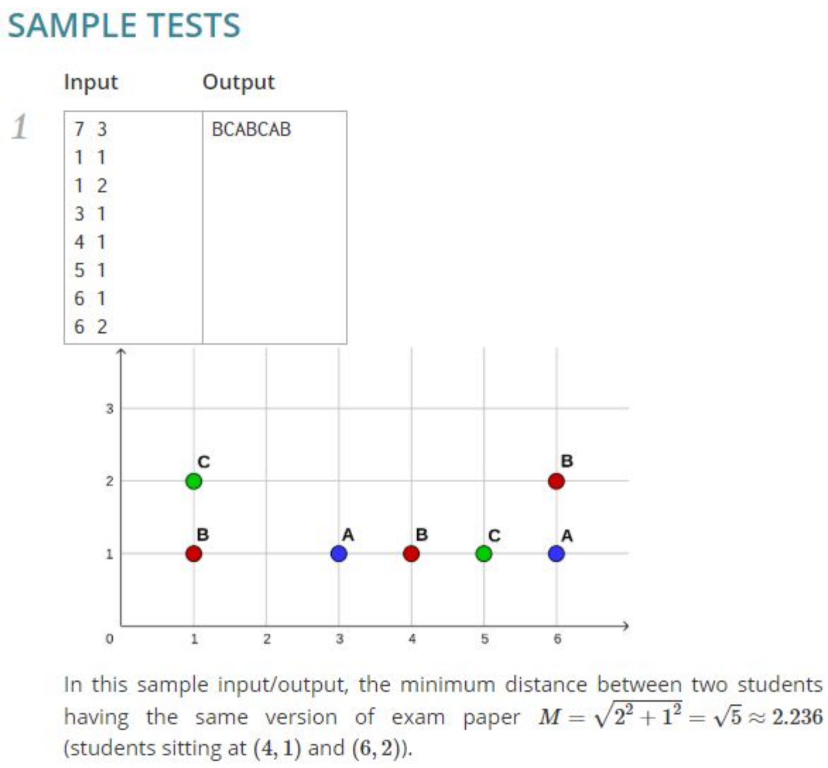
T183 Exam Anti-Cheat

Given a set of **N** points `(x[i], y[i])`

- `0 <= x[i], y[i] <= 1000`
- `x[i], y[i]` are integers

Using **V** colors to color the points, such that:

- for the closest pair of points having same
- their distance is maximized



SAMPLE TESTS

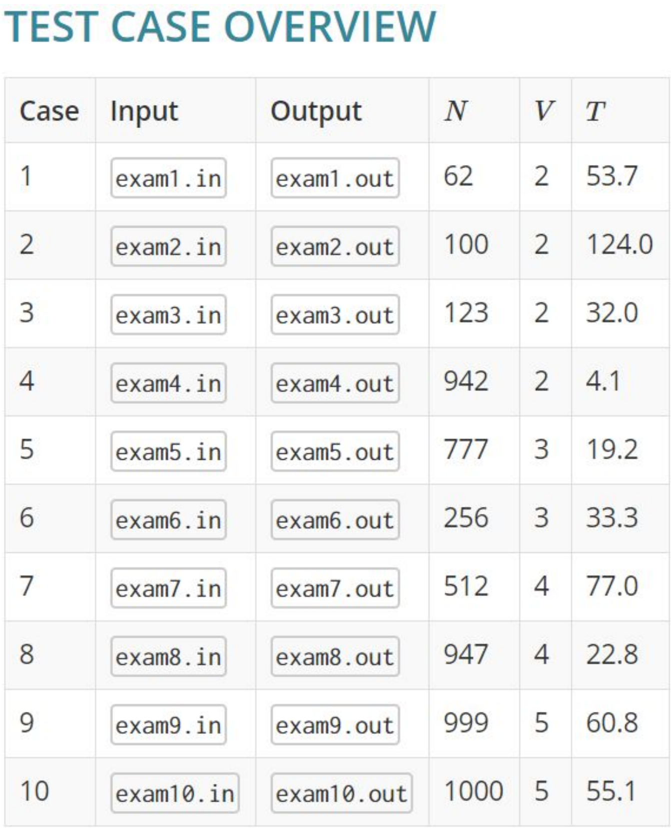
	Input	Output

1	7 3
	1 1
	1 2
	3 1
	4 1
	5 1
	6 1
	6 2
		BCABCAB

In this sample input/output, the minimum distance between two students having the same version of exam paper $M = \sqrt{2^2 + 1^2} = \sqrt{5} \approx 2.236$ (students sitting at $(4, 1)$ and $(6, 2)$).

T183 Exam Anti-Cheat

$10^{(M-D)/(T-D)}$ points for each of the 10 test cases



TEST CASE OVERVIEW

Case	Input	Output	N	V	T
1	`exam1.in`	`exam1.out`	62	2	53.7
2	`exam2.in`	`exam2.out`	100	2	124.0
3	`exam3.in`	`exam3.out`	123	2	32.0
4	`exam4.in`	`exam4.out`	942	2	4.1
5	`exam5.in`	`exam5.out`	777	3	19.2
6	`exam6.in`	`exam6.out`	256	3	33.3
7	`exam7.in`	`exam7.out`	512	4	77.0
8	`exam8.in`	`exam8.out`	947	4	22.8
9	`exam9.in`	`exam9.out`	999	5	60.8
10	`exam10.in`	`exam10.out`	1000	5	55.1

Case #1

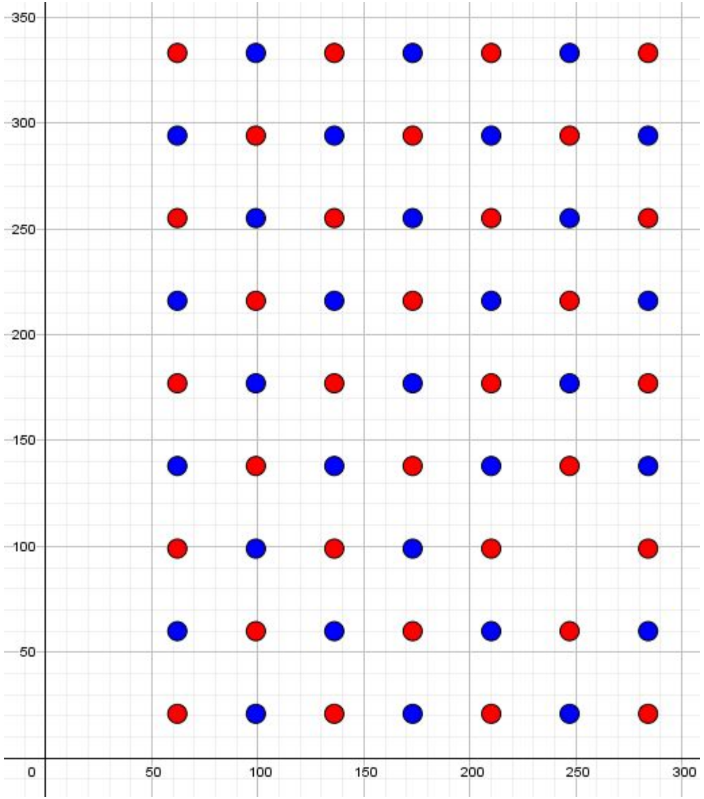
N = 62

V = 2

T = 53.7

D = 37.000

Trainer’s best = 53.759



Case #3

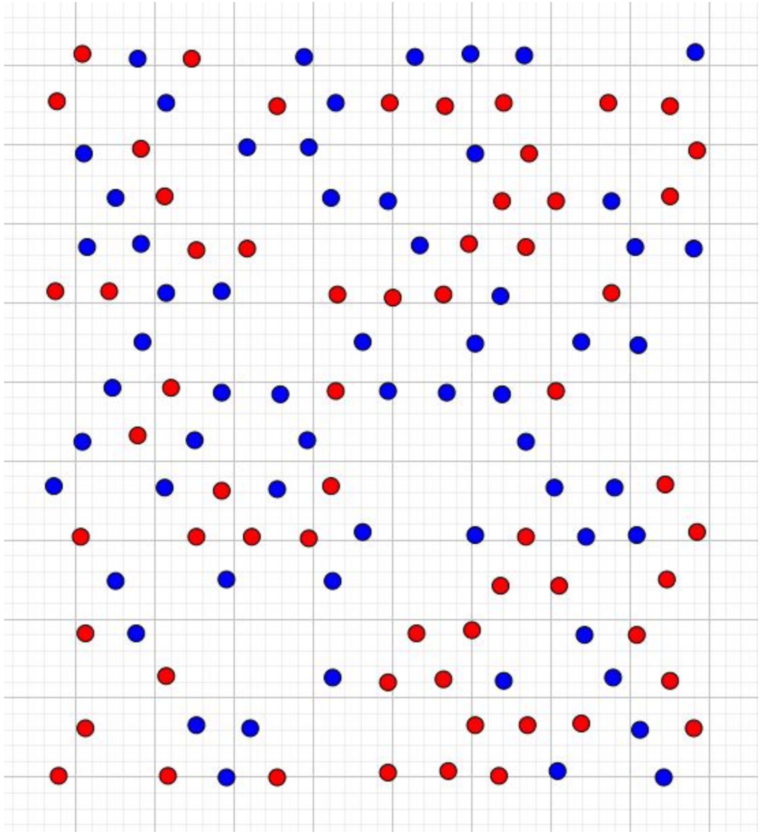
N = 123

V = 2

T = 32.0

D = 30.529

Trainer’s best = 32.016



Case #4

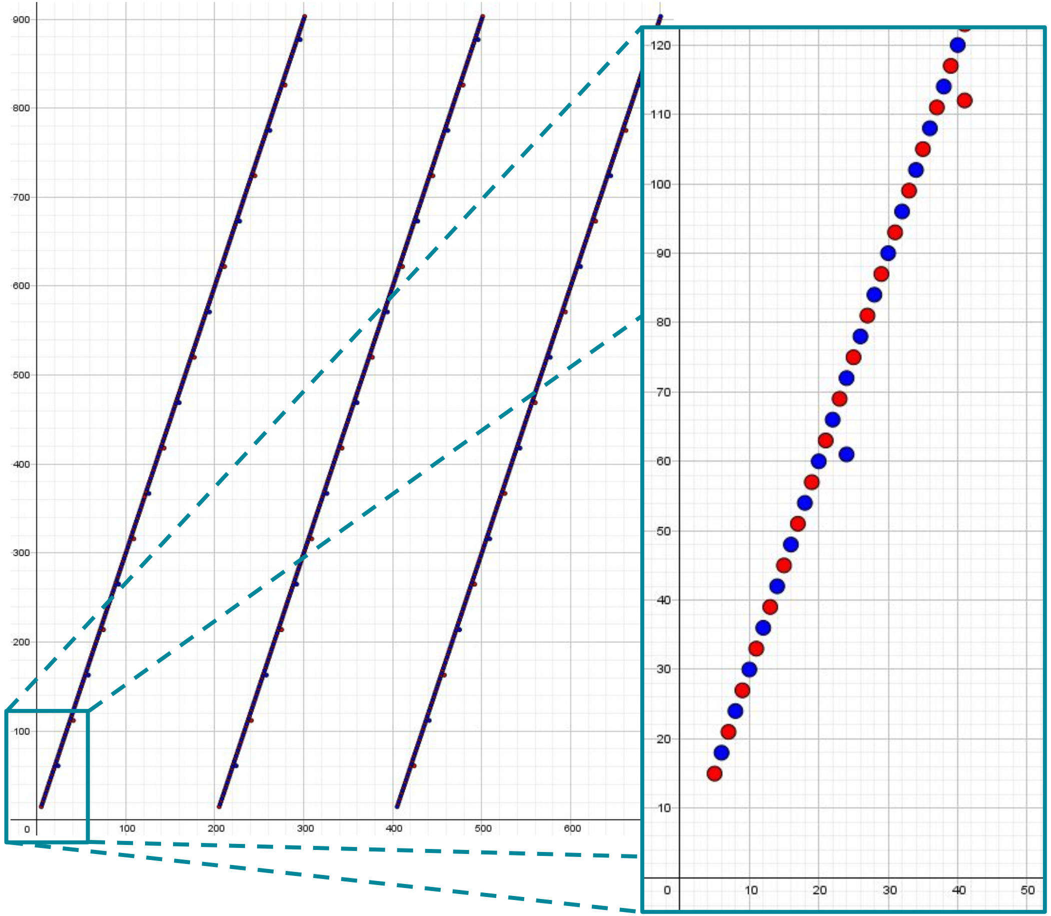
N = 942

V = 2

T = 4.1

D = 3.162

Trainer’s best = 4.123



Case #5

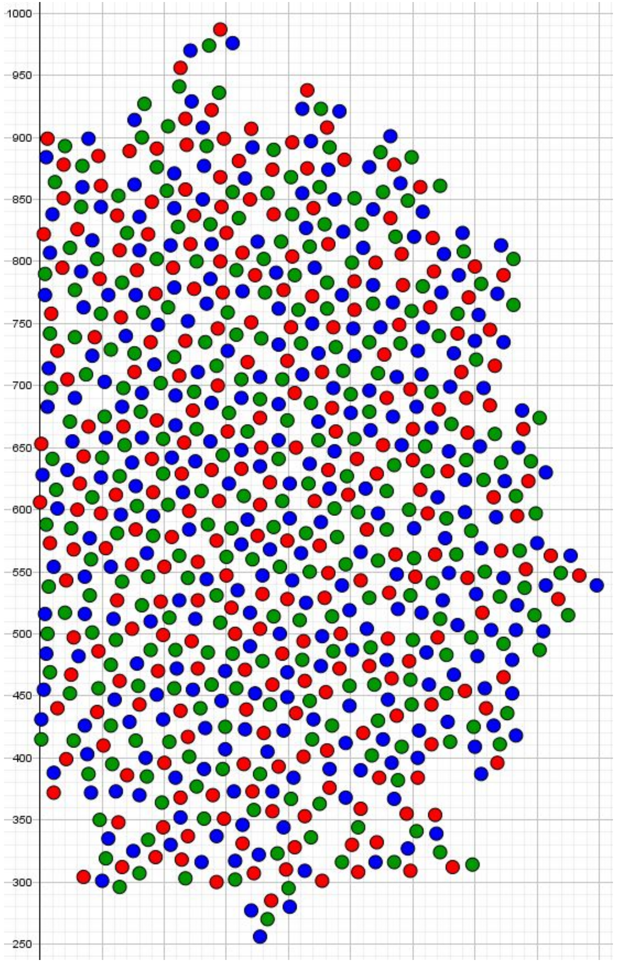
N = 777

V = 3

T = 19.2

D = 14.765

Trainer's best = 19.209



Case #7

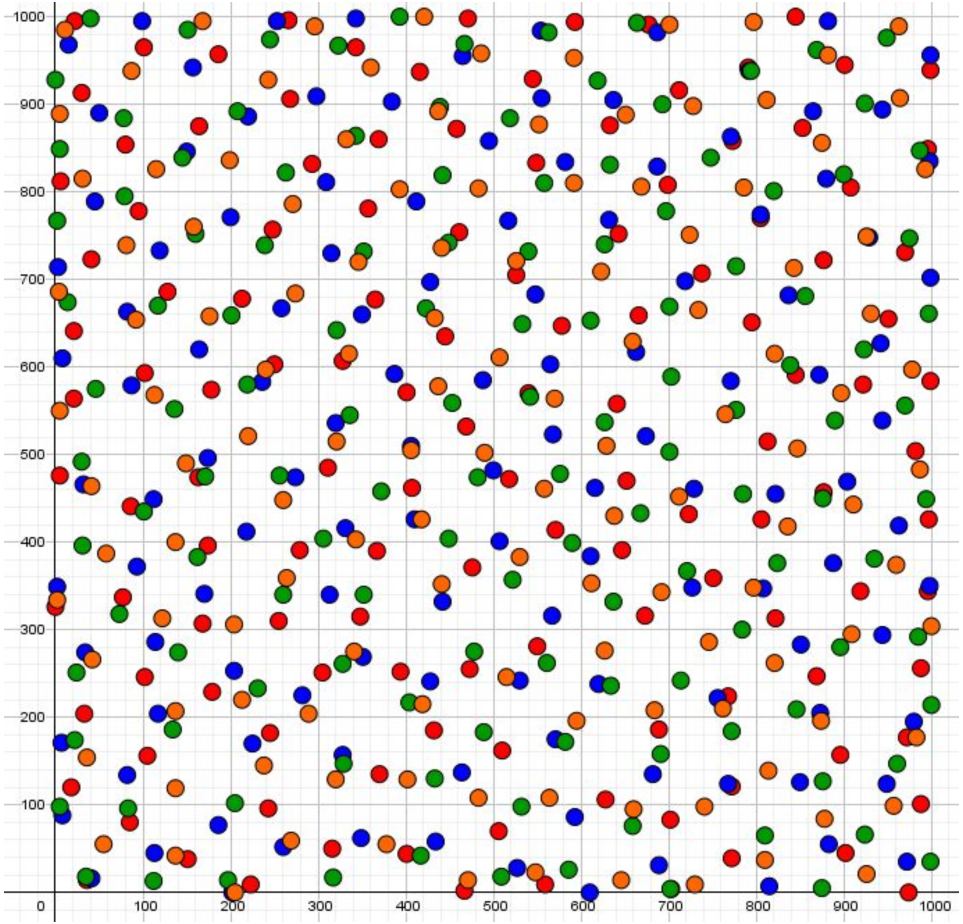
N = 512

V = 4

T = 77.0

D = 1.000

Trainer’s best = 77.000



Solution for 10 points

- $10^{(M-D)/(T-D)}$ points for each of the 10 test cases
- meaning if you get M = D, you score 1 point in each test case
- as D is the minimum distance of a pair in the input
- outputting AAAAAAA... can obtain T = D
- easy 10 points :)

Solution for ~10.5 points

- randomly assign colors to the points

Solution for ~12 points

- outputting the characters periodically, i.e. `ABCDABCDABCD`…

Solution for ~22 points

- randomly assign colors to the points
- repeat this 10000 times, output the best one

Solution for ~48 points

- you should know that for V = 2, there exists optimal solution
- we can do it by *binary search on answer*
- building edges between points with `dist < mid`
- check if the graph is bipartite or not

Few things learnt from T183

There must be some easy points

- Just outputting AAAAAAA...
  - gives you 10 points
- Randomly assign colors and check, repeat many times and output the best
  - gives you 22 points or even more :)

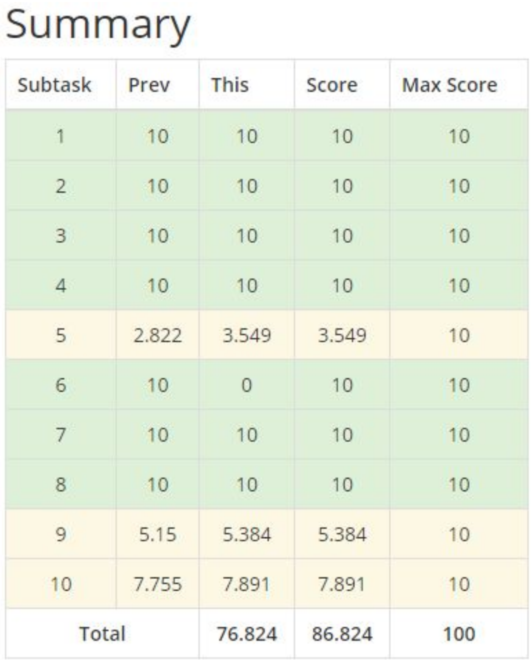
You can try to remodel the problem

- For V = 2, it’s actually a standard problem with optimal solution
  - Remodel the input as a graph with **N** nodes and **N(N-1)** weighted edges
  - gives you 48 points

Few things learnt from T183

Keep running an exhaustion program (perhaps with some optimizations)

- Submit when points increase

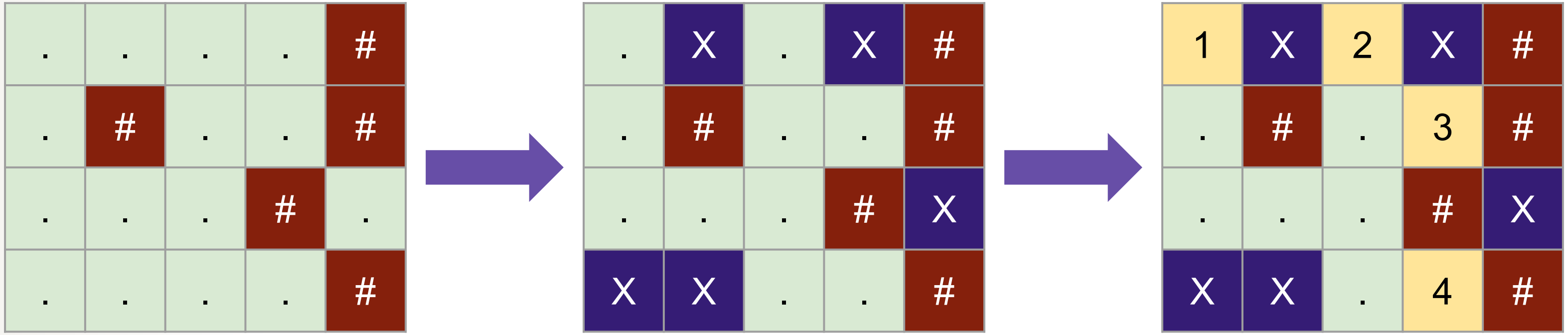


Summary

Subtask	Prev	This	Score	Max Score
1	10	10	10	10
2	10	10	10	10
3	10	10	10	10
4	10	10	10	10
5	2.822	3.549	3.549	10
6	10	0	10	10
7	10	10	10	10
8	10	10	10	10
9	5.15	5.384	5.384	10
10	7.755	7.891	7.891	10
Total		76.824	86.824	100

I1711 Nowruz

- Given an **n** × **m** grid with some obstacle cells
- Build a maze that has as many「堀頭路」(dead end) as possible
  - 「堀頭路」(dead end): cell that has exactly one free neighbour



.	.	.	.	#
.	#	.	.	#
.	.	.	#	.
.	.	.	.	#

.	X	.	X	#
.	#	.	.	#
.	.	.	#	X
X	X	.	.	#

1	X	2	X	#
.	#	.	3	#
.	.	.	#	X
X	X	.	4	#

Techniques

- Try to come up with as many ideas as possible, some ideas may work better than you think
  - Could be weak inputs, weak scoring function or inherent limitation of the problem
  - Assess which idea is the most cost-effective (cost = coding time)
- Visualization
  - If provided, you can use a spreadsheet program to make charts
  - Write a program to generate svg graphics and view them via a web browser

(from T174 - Constellation solution PPT)

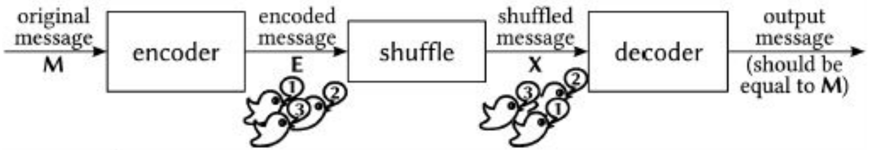
Communication task (Two-steps task)

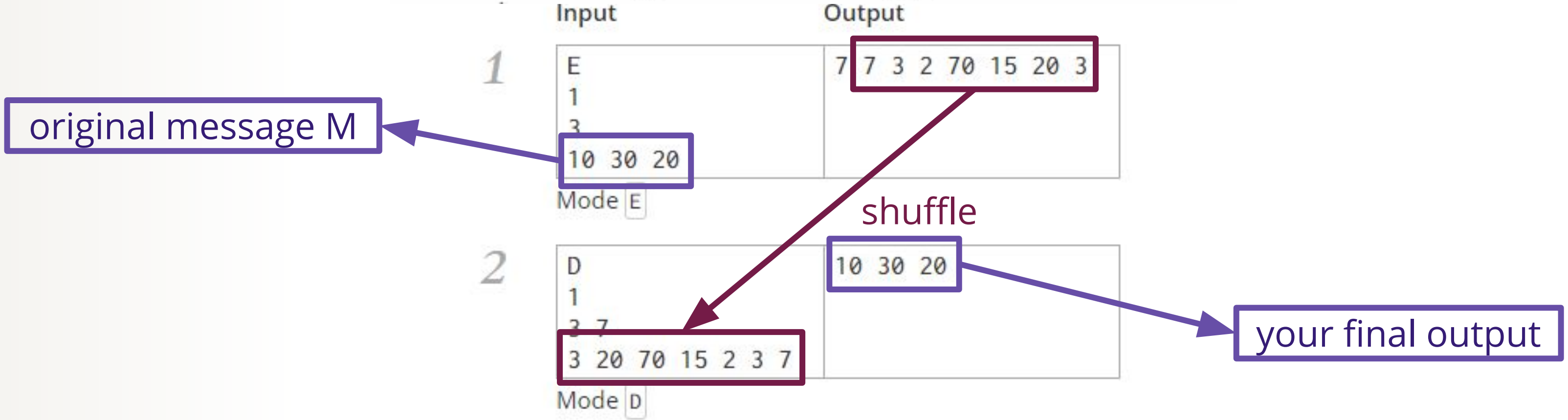
- You have to write two subprograms (or two modes)
- Judging flow:
  - [source input] ➔ [program mode A] ➔ [output A]
  - [input based on output A] ➔ [program mode B] ➔ [final output]
- Your score usually depends on the length / efficiency of [output A]

- Program mode A
  - build up a strategy that can transfer more information with shorter length
- Program mode B
  - build up a strategy to interpret the [output A] and extract some useful data

I1123 Parrots

- Original message **M** consists of at most 64 integers within [0, 255]





	Input	Output
1	E 1 3 10 30 20	7 7 3 2 70 15 20 3
	Mode E	
2	D 1 3 7 3 20 70 15 2 3 7	10 30 20
	Mode D	

original message M

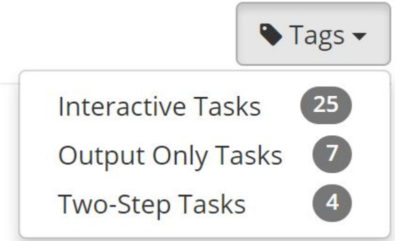
shuffle

your final output

Conclusion

- Just like constructive task, non-batch task is another type of problems
  - NOT LIMITED by any algorithms, topics
  - therefore, no standard rules to deal with them
  - again, “practice makes perfect”
  - as long as you solve / take a look at more non-batch tasks,
  - more techniques / experiences you can accumulate
- From the history of Team Formation Test,
  - non-batch tasks often appear :)
  - good luck :)

Practice Tasks



Interactive Tasks 25
Output Only Tasks 7
Two-Step Tasks 4

- [IOI2013] Cave [IOI2014] Game [IOI2015] Scales
- [IOI2016] Unscrambling a Messy Bug [IOI2017] The Big Prize
- [IOI2018] Highway Tolls [IOI2018] Combo
- [IOI2010] Maze [IOI2012] Pebbling Odometer [IOI2017] Nowruz
- [IOI2011] Parrots

- [TFT2012] Debug! [TFT2013] The Forgotten Triangle
- [TFT2016] Model Answer II [TFT2018] Cave Exploration
- [TFT2017] Constellation [TFT2018] Exam Anti-Cheat
- [TFT2014] Lost Sequence

● **Interactive** ● **Output-only** ● **Two-steps**