Sets

1. What is a set

Set in Python is a data structure equivalent to sets in mathematics. It may consist of various elements; the order of elements in a set is undefined. You can add and delete elements of a set, you can iterate the elements of the set, you can perform standard operations on sets (union, intersection, difference). Besides that, you can check if an element belongs to a set.

Unlike arrays, where the elements are stored as ordered list, the order of elements in a set is undefined (moreover, the set elements are usually not stored in order of appearance in the set; this allows checking if an element belongs to a set faster than just going through all the elements of the set).

Any immutable data type can be an element of a set: a number, a string, a tuple. Mutable (changeable) data types cannot be elements of the set. In particular, list cannot be an element of a set (but tuple can), and another set cannot be an element of a set. The requirement of immutability follows from the way how do computers represent sets in memory.

2. How to define a set

You can define a set as simple as by naming all of its elements in brackets. The only exception is *empty set*, which can be created using the function `set()`. If `set(..)` has a list, a string or a tuple as a parameter, it will return a set composed of its elements. For example,

```
A = {1, 2, 3}

A = set('qwerty')

print(A)
```

will print `{'e', 'q', 'r', 't', 'w', 'y'}` as the output.

The order of elements is unimportant. For example, the program

```
A = {1, 2, 3}

B = {3, 2, 3, 1}

print(A == B)
```

will print `True`, because `A` and `B` are equal sets.

Each element may enter the set only once. `set('Hello')` returns the set of four elements: `{'H', 'e', 'l', 'o'}`.

3. Operations with elements

You can get the number of elements in the set using the function `len`.

You can also iterate over all the elements of the set (in an undefined order!) using the loop `for`:

```
primes = {2, 3, 5, 7, 11}

for num in primes:

    print(num)
```

You can check whether an element belongs to a set using the keyword `in`: expressions like `a in A` return a value of type `bool`. Similarly there's the opposite operation `not in`. To add an element to the set there is the method `add`:

```
A = {1, 2, 3}

print(1 in A, 4 not in A)

A.add(4)
```

There are two methods to remove an element from a set: `discard` and `remove`. Their behavior varies only in case if the deleted item was not in the set. In this case the method `discard` does nothing and the method`remove` throws exception `KeyError`.

Finally, `pop` removes one random element from the set and returns its value. If the set is empty, `pop`generates the exception `KeyError`.

You can transform a set to list using the function `list`.

4. Operations on sets

This is how you perform the well-known operations on sets in Python:

A | B **A.union(B)**	Returns a set which is the union of sets `A` and `B`.
A |= B **A.update(B)**	Adds all elements of array `B` to the set `A`.
A & B **A.intersection(B)**	Returns a set which is the intersection of sets `A` and `B`.
A &= B **A.intersection_update(B)**	Leaves in the set `A` only items that belong to the set `B`.
A - B **A.difference(B)**	Returns the set difference of `A` and `B` (the elements included in `A`, but not included in `B`).
A -= B **A.difference_update(B)**	Removes all elements of `B` from the set `A`.
A ^ B **A.symmetric_difference(B)**	Returns the symmetric difference of sets `A` and `B` (the elements belonging to either `A` or `B`, but not to both sets simultaneously).
A ^= B **A.symmetric_difference_update(B)**	Writes in `A` the symmetric difference of sets `A` and `B`.
A <= B **A.issubset(B)**	Returns `true` if `A` is a subset of `B`.
A >= B **A.issuperset(B)**	Returns `true` if `B` is a subset of `A`.

A < B	Equivalent to `A <= B and A != B`
A > B	Equivalent to `A >= B and A != B`

Problem «The number of distinct numbers» (Easy)

Statement

Given a list of integers. Determine how many distinct numbers there are.

This task can be solved in one line of code.

Your solution

```
print(len({int(s) for s in input().split()}))
```

Suggested solution

```
print(len(set(input().split())))
```

Problem «The number of equal numbers» (Easy)

Statement

Given two lists of numbers. Count how many unique numbers occur in both of them.

This task can be solved in one line of code.

Your solution

```
a = {int(s) for s in input().split()}

b = {int(d) for d in input().split()}

print(len(a.intersection(b)))
```

Suggested solution

```
print(len(set(input().split()) & set(input().split())))
```

Problem «The intersection of sets» (Medium)

Statement

Given two lists of numbers. Find all the numbers that occur in both the first and the second list and print them in ascending order.

Even this task can be solved in one line of code.

Your solution

```
D = set(input().split()).intersection(set(input().split()))

D = sorted(D)

for num in D:

    print(num, end= " ")
```

Suggested solution

```
print(*sorted(set(input().split()) & set(input().split()), key=int))
```

Problem «Has the number been encountered before» (Medium)

Statement

Given a sequence of numbers, determine if the next number has already been encountered. For each number, print the word `YES` (in a separate line) if this number has already been encountered, and print `NO`, if it has not already been encountered.

Your solution

```
l = [int(i) for i in input().split()]

s = set()

for i in range(len(l)):

    if l[i] not in s:

        print('NO')

        n = l[i]

        b = {n}

        s = s.union(b)

    else:

        print('YES')
```

Suggested solution

```
numbers = [int(s) for s in input().split()]

occur_before = set()

for num in numbers:

    if num in occur_before:

        print('YES')

    else:
```

```
    print('NO')
    occur_before.add(num)
```

Problem «Cubes» (Medium)

Statement

Alice and Bob like to play with colored cubes. Each child has its own set of cubes and each cube has a distinct color, but they want to know how many unique colors exist if they combine their block sets. To determine this, the kids enumerated each distinct color with a random number from 00 to 10_8108. At this point their enthusiasm dried up, and you are invited to help them finish the task.

Given two integers that indicate the number of blocks in Alice's and then Bob's sets NN and MM. The following NN lines contain the numerical color value for each cube in Alice's set. Then the last MM rows contain the numberical color value for each cube in Bob's set.

Find three sets: the numerical colors of cubes in both sets, the numerical colors of cubes only in Alice's set, and the numerical colors of cubes only in Bob's set. For each set, print the number of elements in the set, followed by the numerical color elements, sorted in ascending order.

Your solution

```
C = [int(s) for s in input().split()]
N = C[0]
M = C[1]
a = []
b = []
for i in range(N):
   a.append(int(input()))
for i in range(M):
   b.append(int(input()))
A = set(a)
B = set(b)
s1 = sorted(A.intersection(B))
s2 = sorted(A.difference(B))
s3 = sorted(B.difference(A))
print(len(s1))
for num in s1:
   print(num)
```

```
print(len(s2))
for num in s2:
    print(num)
print(len(s3))
for num in s3:
    print(num)
```

Suggested solution

```
def print_set(some_set):
    print(len(some_set))
    print(*[str(item) for item in sorted(some_set)])
N, M = [int(s) for s in input().split()]
A_colors, B_colors = set(), set()
for i in range(N):
    A_colors.add(int(input()))
for i in range(M):
    B_colors.add(int(input()))


print_set(A_colors & B_colors)
print_set(A_colors - B_colors)
print_set(B_colors - A_colors)
```

Problem «The number of distinct words in some text» (Medium)

Statement

Given a number n, followed by n lines of text, print the number of distinct words that appear in the text.

For this, we define a word to be a sequence of non-whitespace characters, seperated by one or more whitespace or newline characters. Punctuation marks are part of a word, in this definition.

Your solution

```
n = int(input())
s = {str(i) for i in input().split()}
for i in range(n-1):
```

```
    b = {str(i) for i in input().split()}
    s = s.union(b)
print(len(s))
```

Suggested solution

```
words = set()
for _ in range(int(input())):
    words.update(input().split())
print(len(words))
```

Problem «Guess the number» (Medium)

Statement

Augustus and Beatrice play the following game. Augustus thinks of a secret integer number from 1 to n. Beatrice tries to guess the number by providing a set of integers. Augustus answers `YES` if his secret number exists in the provided set, or `NO`, if his number does not exist in the provided numbers. Then after a few questions Beatrice, totally confused, asks you to help her determine Augustus's secret number.

Given the value of n in the first line, followed by the a sequence Beatrice's guesses, series of numbers seperated by spaces and Agustus's responses, or Beatrice's plea for `HELP`. When Beatrice calls for help, provide a list of all the remaining possible secret numbers, in ascending order, separated by a space.

Your solution

```
n = int(input())
G1 = set(input().split())
ANS = str(input())
G2 = set(input().split())
end = set(str("HELP").split())
while end != G2:
    ANS2 = str(input())
    if ANS == 'YES' and ANS2 == 'YES':
        G1.intersection_update(G2)
        ANS = ANS2
    if ANS == 'YES' and ANS2 == 'NO':
        G1.difference_update(G2)
        ANS = ANS2
```

```
    if ANS == 'NO' and ANS2 == 'NO':
        G1.difference_update(G2)
        ANS = ANS2
    G2 = set(input().split())
else:
    G1 = sorted(G1)
    for elem in G1:
        print(elem, end= ' ')
```

Suggested solution

```
n = int(input())
all_nums = set(range(1, n + 1))
possible_nums = all_nums
while True:
    guess = input()
    if guess == 'HELP':
        break
    guess = {int(x) for x in guess.split()}
    answer = input()
    if answer == 'YES':
        possible_nums &= guess
    else:
        possible_nums &= all_nums - guess
print(' '.join([str(x) for x in sorted(possible_nums)]))
```

Problem «Polyglots» (Hard)

Statement

Each student at a certain school speaks a number of languages. We need to determine which languges are spoken by all the students, which languages are spoken by at least one student.

Given, the number of students, and then for each student given the number of languages they speak followed by the name of each language spoken, find and print the number of languages spoken by all the students, followed by a list the languages by name, then print the number of languages spoken by at least one student, followed by the list of the languages by name. Print the languages in alphabetical order.

Your solution

```
n = int(input())
l = int(input())
L = set()
for i in range(l):
    S = set(input().split())
    L = L.union(S)
for j in range(n-1):
    l = int(input())
    B = set()
    for i in range(l):
        S = set(input().split())
        B = B.union(S)
    D = L.intersection(B)
D = sorted(D)
L = sorted(L)
print(len(D))
for elem in D:
    print(elem)
print(len(L))
for elem in L:
    print(elem)
```

Suggested solution

```
students = [{input() for j in range(int(input()))} for i in range(int(input()))]
known_by_everyone, known_by_someone = set.intersection(*students), set.union(*students)
print(len(known_by_everyone), *sorted(known_by_everyone), sep='\n')
print(len(known_by_someone), *sorted(known_by_someone), sep='\n')
```