How to remove elements from set?

1. remove()
2. clear()
3. discard()
4. pop()

remove(element)

This method remove given element from set if exists, if given element not exists it raises KeyError

```
>>> A={10,20,30,40,50}
>>> print(A)
{50, 20, 40, 10, 30}
>>> A.remove(10)
>>> print(A)
{50, 20, 40, 30}
>>> A.remove(10)
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    A.remove(10)
KeyError: 10
```

discard(element)

This method remove given element from set if exists, if given element is not exists it will not raise any error.

```
>>> A={10,20,30,40,50}
>>> print(A)
{50, 20, 40, 10, 30}
>>> A.discard(30)
>>> print(A)
{50, 20, 40, 10}
>>> A.discard(30)
```

pop()
This method removes an arbitrary element from set. Before removing it returns removed element.

```
>>> B={10,20,30,40,50}
>>> print(B)
{50, 20, 40, 10, 30}
>>> value1=B.pop()
>>> print(B)
{20, 40, 10, 30}
>>> print(value1)
50
>>> value2=B.pop()
>>> print(value2)
20
```

clear()
This method removes all the elements from set

```
>>> A=set(range(10,110,10))
>>> print(A)
{100, 70, 40, 10, 80, 50, 20, 90, 60, 30}
>>> A.clear()
>>> print(A)
set()
```

https://www.hackerrank.com/challenges/py-set-discard-remove-pop/problem?isFullScreen=false

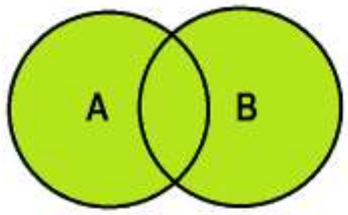
```
n=int(input())
A=set(map(int,input().split()))
N=int(input())

for i in range(N):
    cmd=input().split()
    if cmd[0]=="pop":
        A.pop()
    elif cmd[0]=="discard":
        A.discard(int(cmd[1]))
    elif cmd[0]=="remove":
        A.remove(int(cmd[1]))

print(sum(A))
```

Set Operations
Advantage of using sets is performing mathematical set operations.
Union, intersection, difference,….

union(*others)
set | other | ...
Return a new set with elements from the set and all others.



A.union(B) or A|B

```
>>> A={1,2,3}
>>> B={4,5,6}
>>> C=A.union(B)
>>> print(A,B,C,sep="\n")
{1, 2, 3}
{4, 5, 6}
{1, 2, 3, 4, 5, 6}
>>> D=[6,7,8]
>>> X=A.union(D)
>>> print(X)
{1, 2, 3, 6, 7, 8}
>>> A={'A','B','C','D'}
>>> B={'E','F','G','H'}
>>> C=A.union(B)
>>> print(A,B,C,sep="\n")
```

```
{'B', 'D', 'A', 'C'}
{'G', 'E', 'H', 'F'}
>>> A.union("ABC")
{'B', 'D', 'A', 'C'}
```

https://www.hackerrank.com/challenges/py-set-union/problem?isFullScreen=false

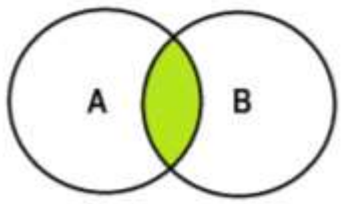
```
n=int(input())
E=set(map(int,input().split()))
b=int(input())
F=set(map(int,input().split()))

print(len(E.union(F)))
```

Example:

```
>>> A={1,2,3}
>>> B={1,2,4}
>>> C={2,3,4,5}
>>> D=A|B|C
>>> print(A,B,C,D,sep="\n")
{1, 2, 3}
{1, 2, 4}
{2, 3, 4, 5}
{1, 2, 3, 4, 5}
```

intersection(*others*)
set & other & ...
Return a new set with elements common to the set and all others.

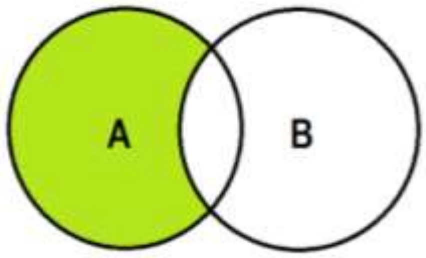


A.intersection(B) or A&B

```
>>> python_students={"naresh","suresh","ramesh"}
>>> java_students={"kishore","rajesh","suresh"}
>>>
python_java_students=python_students.intersection(java_students)
>>> print(python_students)
{'ramesh', 'naresh', 'suresh'}
>>> print(java_students)
{'kishore', 'rajesh', 'suresh'}
>>> print(python_java_students)
{'suresh'}

>>> A=10
>>> B=20
>>> c=A&B
>>> print(bin(A),bin(B),bin(c))
0b1010 0b10100 0b0
>>> A={1,2,3,4,5}
>>> B={1,2,3,6,7}
>>> C=A&B
>>> print(A,B,C,sep="\n")
{1, 2, 3, 4, 5}
{1, 2, 3, 6, 7}
{1, 2, 3}
```

difference(*others*)
set - other - ...
Return a new set with elements in the set that are not in the others.



A.difference(B) or A - B

```
>>> A={1,2,3,4,5}
>>> B={1,2,3,6,7}
>>> C=A.difference(B)
>>> print(A,B,C,sep="\n")
{1, 2, 3, 4, 5}
{1, 2, 3, 6, 7}
{4, 5}
```

https://www.hackerrank.com/challenges/py-set-difference-operation/problem?isFullScreen=false

```
n=int(input())
E=set(map(int,input().split()))
b=int(input())
F=set(map(int,input().split()))
print(len(E-F))
```

symmetric_difference(*other*)
set ^ other
Return a new set with elements in either the set or *other* but not both.

```
>>> A={1,2,3,4,5}
>>> B={1,2,3,6,7}
>>> C=A.symmetric_difference(B)
>>> print(A,B,C,sep="\n")
{1, 2, 3, 4, 5}
{1, 2, 3, 6, 7}
{4, 5, 6, 7}
```

https://www.hackerrank.com/challenges/symmetric-difference/problem?isFullScreen=false

```
n=int(input())
A=set(map(int,input().split()))
b=int(input())
B=set(map(int,input().split()))
C=A^B
```

```
D=sorted(C)
for value in D:
    print(value)
```

update(**others*)
set |= other | ...
Update the set, adding elements from all others.

```
>>> A={1,2,3,4}
>>> A.update({1,2,3,4,5,6,7})
>>> print(A)
{1,2,3,4,5,6,7}
>>> A={1,2,3}
>>> B={4,5,6}
>>> A|=B
>>> print(A)
{1, 2, 3, 4, 5, 6}
```

intersection_update(**others*)
set &= other & ...
Update the set, keeping only elements found in it and all others.

```
>>> A={1,2,3,4,5}
>>> B={1,2,6,7,8}
>>> A.intersection_update(B)
>>> print(A)
{1, 2}
```

difference_update(**others*)
set -= other | ...
Update the set, removing elements found in others.

```
>>> A={1,2,3,4,5}
>>> B={1,2,3}
>>> A.difference_update(B)
>>> print(A)
```

```
{4, 5}
```

symmetric_difference_update(*other*)
set ^= other
Update the set, keeping only elements found in either set, but not in both.

```
>>> A={1,2,3,4}
>>> B={1,2,5,6}
>>> A^=B
>>> print(A)
{3, 4, 5, 6}
```

Set examine methods

1. issuperset(*other*)

set >= other
Test whether every element in *other* is in the set.

```
>>> A={1,2,3,4,5}
>>> B={1,2,3,6,7}
>>> A>B
False
>>> A>=B
False
>>> C={1,2,3,4,5}
>>> A.issuperset(C)
True
>>> D={1,2,3}
>>> A>=D
True
```

2. issubset(*other*)

set <= other
Test whether every element in the set is in *other*.

```
>>> A={1,2,3,4,5}
>>> B={1,2,3}
>>> A.issubset(B)
False
>>> C={1,2,3,4,5,6,7}
>>> A.issubset(C)
True
```

3. isdisjoint(*other*)

Return True if the set has no elements in common with *other*. Sets are disjoint if and only if their intersection is the empty set.

```
>>> A={1,2,3}
>>> B={4,5,6}
>>> A.isdisjoint(B)
True
>>> C={1,6,7}
>>> A.isdisjoint(C)
False
```