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

```
n=int(input())
A=set(map(int,input().split()[:n]))
N=int(input())
for i in range(N):
    cmd=input().split() # update 5 # ["update","5"]
    B=set(map(int,input().split()[:int(cmd[1])]))
    if cmd[0]=="update":
        A.update(B)
    elif cmd[0]=="difference_update":
        A.difference_update(B)
    elif cmd[0]=="intersection_update":
        A.intersection_update(B)
    elif cmd[0]=="symmetric_difference_update":
        A.symmetric_difference_update(B)

print(sum(A))
```

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={10,20,30,40,50}
>>> B={60,70,80,90,100}
>>> A.isdisjoint(B)
True
>>> python_student={"naresh","suresh"}
>>> java_student={"kishore","kiran"}
>>> python_student.isdisjoint(java_student)
True
```

issubset(*other*)
set <= other
Test whether every element in the set is in *other*
issuperset(*other*)
set >= other
Test whether every element in *other* is in the set.

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

frozenset

frozenset is an immutable set. After creating frozenset we cannot add and remove objects. Frozenset does not provide mutable methods like,

1. add()
2. remove()
3. pop()
4. clear()
5. update()
6. difference_update()
7. intersection_update()
8. symmetric_difference_update()

What is use of frozneset?

1. For representing immutable set
2. For representing set inside set (nested set)

How to create forznenset?
This frozenset is created using,

1. frozentset()
2. frozenset(iterable)

```
>>> set1=frozenset()
>>> set1.add(10)
Traceback (most recent call last):
  File "<pyshell#14>", line 1, in <module>
    set1.add(10)
AttributeError: 'frozenset' object has no attribute 'add'
>>> set1=frozenset(range(10,60,10))
>>> print(set1)
>>> frozenset({40, 10, 50, 20, 30})
>>> set2=frozenset({10,20,30,40,50})
print(set2)
frozenset({50, 20, 40, 10, 30})
>>> set3={frozenset({1,2,3}),frozenset({4,5,6})}
>>> print(set3)
{frozenset({1, 2, 3}), frozenset({4, 5, 6})}
>>> for s in set3:
...     print(s)
...
...
frozenset({1, 2, 3})
frozenset({4, 5, 6})
```

What is difference between list and set?

List	**Set**
List is ordered collection	Set is unordered collection
List support index and slicing	Set does not support indexing and slicing
List allows duplicates values	Set does not allows duplicate values
List allows any type of objects	Allows only hashable objects
In List elements are organized in sequential order	In set elements are organized using hashing data structure
List is create using []	Set is created using {}
"list" class or data type is used for representing list object	"set" class or data type is used for representing set object
In application development list is used to represent group of individual objects where insertion order is preserved and allows to read sequential and randomly and allows duplicates.	In application development set is used to represent group of objects where duplicate not allowed and perform some mathematical set operations

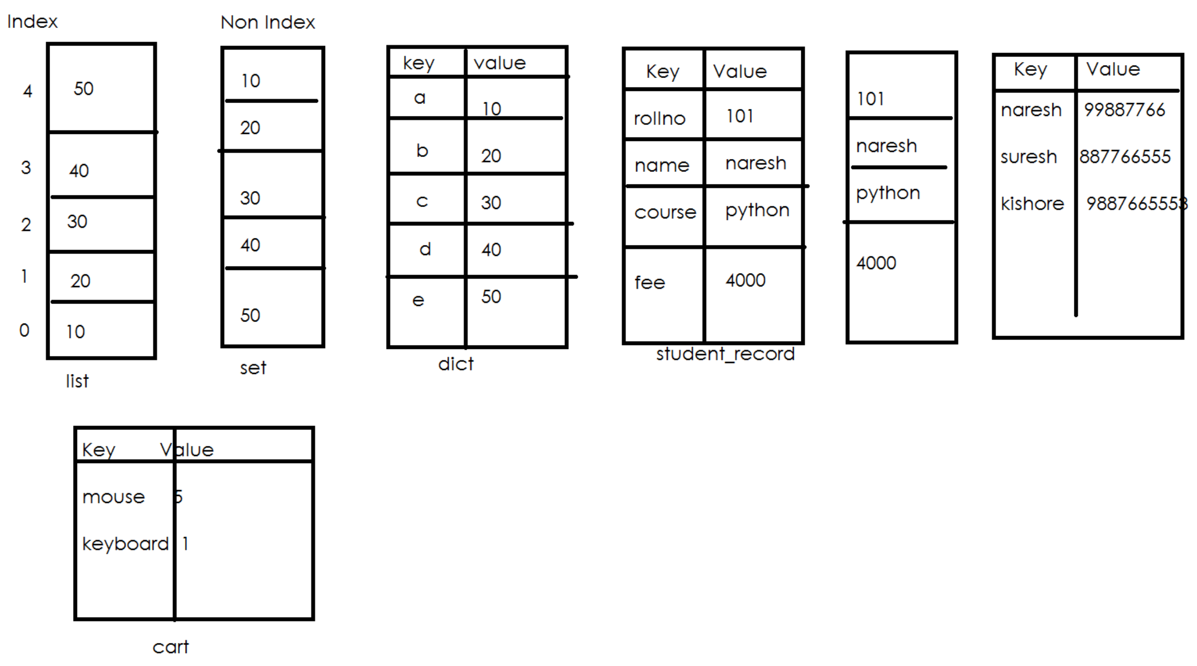
Dictionary or mapping

"dict" data type or class used to represent dictionary object.
In dictionary data is organized as pair of values.

1. Key

2. Value

Each value in dictionary is identified with key
One key is mapped with one or more than one value.
Dictionary is key based collection.
Dictionary is mutable collection.
Dictionary does not allow duplicate keys but duplicate values are allowed.
Dictionary keys are immutable and values are mutable.



How to create dictionary?

Dictionaries can be created by several means:

- Use a comma-separated list of key: value pairs within braces: {'jack': 4098, 'sjoerd': 4127} or {4098: 'jack', 4127: 'sjoerd'}

Example:

```
>>> dict1={}
>>> type(dict1)
<class 'dict'>
```

```
>>> dict1={1:10,2:20,3:30,4:40,5:50}
>>> print(dict1)
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}
>>> dict2={1:10,1:20,1:30,1:40,1:50}
>>> print(dict2)
{1: 50}
>>> dict3={1:10,2:10,3:10,4:10,5:10}
>>> print(dict3)
{1: 10, 2: 10, 3: 10, 4: 10, 5: 10}
```

- **Use a dict comprehension: {}, {x: x ** 2 for x in range(10)}**
- Use the type constructor: dict(), dict([('foo', 100), ('bar', 200)]), dict(foo=100, bar=200)

Dictionary is represented in curly braces { }

Set → list()
List → set()
Dict → set()
Dict → list()
List → tuple()
Range → list()