Department :- Computer Science & Engineering **Class :-** TY-B Batch :- B2

Course Name :- Competitive Programming Lab **Course Code :-** BTCOC606

Name Of The Student :- Kunal Sapkal

Roll No. :- 113 **PRN :-** 2162701242113

Experiment No. 8

Challenge 1:- Collections.OrderedDict()

Challenge 2:- Symmetric Difference

Challenge 3:- Set.add()

Challenge 4:- Set .discard(), .remove() & .pop()

Challenge 5:- Collections.deque()

Challenge 1:- Collections.OrderedDict() :-

An OrderedDict is a dictionary that remembers the order of the keys that were inserted first. If a new entry overwrites an existing entry, the original insertion position is left unchanged.

Examp

le Code

```
from collections import OrderedDict

ordinary_dictionary = {}

ordinary_dictionary['a'] = 1

ordinary_dictionary['b'] = 2

ordinary_dictionary['c'] = 3

ordinary_dictionary['d'] = 4

ordinary_dictionary['e'] = 5

print ordinary_dictionary
```

{'a': 1, 'c': 3, 'b': 2, 'e': 5, 'd': 4}

```
>>> ordered_dictionary = OrderedDict()
>>> ordered_dictionary['a'] = 1
>>> ordered_dictionary['b'] = 2
>>> ordered_dictionary['c'] = 3
>>> ordered_dictionary['d'] = 4
>>> ordered_dictionary['e'] = 5
>>> print ordered_dictionary
OrderedDict([('a', 1), ('b', 2), ('c', 3), ('d', 4), ('e', 5)])
```

Task

You are the manager of a supermarket.
You have a list of N items together with their prices that consumers bought on a particular day.
Your task is to print each item_name and net_price in order of its first occurrence.

item_name = Name of the item.
net_price = Quantity of the item sold multiplied by the price of each item.

Input Format

The first line contains the number of items, N .
The next N lines contains the item's name and price, separated by a space.

Constraints

0<N<=100

Output Format

Print the item_name and net_price in order of its first occurrence.

Sample Input

9

BANANA FRIES 12

POTATO CHIPS 30

APPLE JUICE 10

CANDY 5

APPLE JUICE 10

CANDY 5

CANDY 5

CANDY 5

POTATO CHIPS 30

Sample Output

BANANA FRIES 12

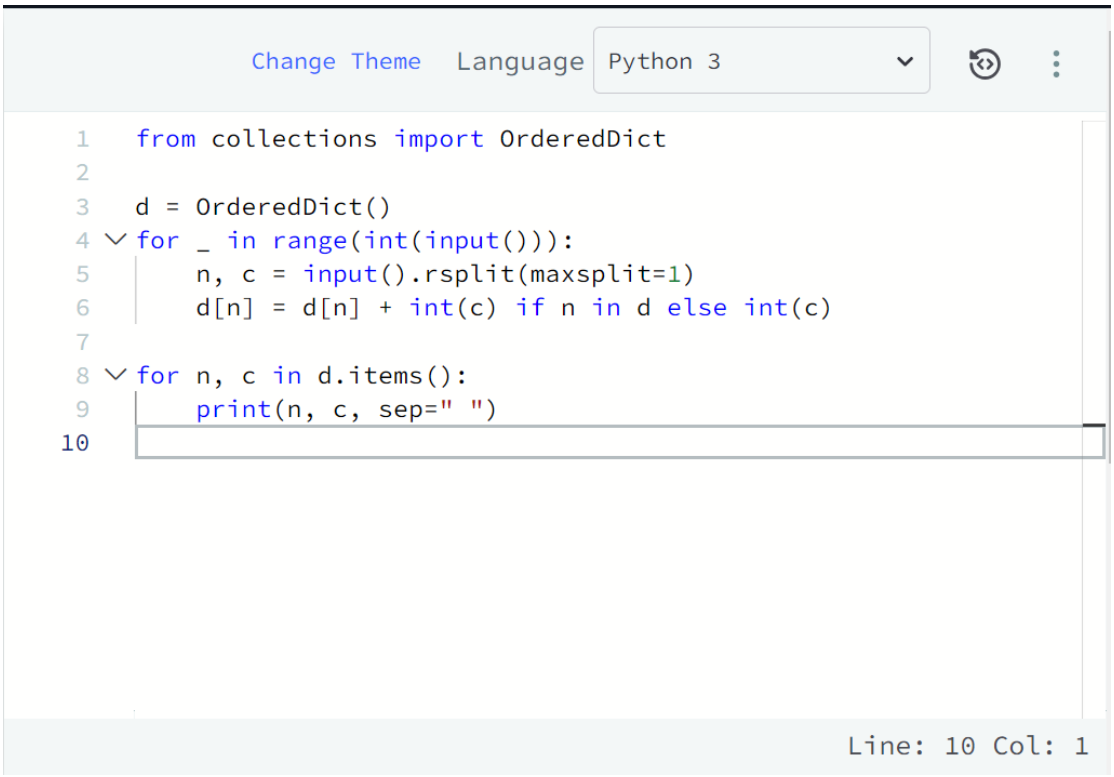
POTATO CHIPS 60

APPLE JUICE 20

CANDY 20

Explanation

BANANA FRIES: Quantity bought:1 , Price: 12
Net Price: 12
POTATO CHIPS: Quantity bought:2 , Price: 30
Net Price: 60
APPLE JUICE: Quantity bought: 2, Price:10
Net Price: 20
CANDY: Quantity bought: 4, Price: 5
Net Price: 20



Challenge 2:- Symmetric Difference :-

Objective

Today, we're learning about a new data type: sets.

Concept

If the inputs are given on one line separated by a character (the delimiter), use split() to get the separate values in the form of a list. The delimiter is space (ascii 32) by default. To specify that comma is the delimiter, use string.split(','). For this challenge, and in general on HackerRank, space will be the delimiter.

Usage:

>> a = raw_input()

5 4 3 2

>> lis = a.split()

>> print (lis)

['5', '4', '3', '2']

If the list values are all integer types, use the map() method to convert all the strings to integers.

>> newlis = list(map(int, lis))

>> print (newlis)

[5, 4, 3, 2]

Sets are an unordered collection of unique values. A single set contains values of any immutable data type.

CREATING SETS

```
>> myset = {1, 2} # Directly assigning values to a set
>> myset = set() # Initializing a set
>> myset = set(['a', 'b']) # Creating a set from a list
>> myset
{'a', 'b'}
```

MODIFYING SETS

Using the add() function:

```
>> myset.add('c')
>> myset
{'a', 'c', 'b'}
>> myset.add('a') # As 'a' already exists in the set, nothing happens
>> myset.add((5, 4))
>> myset
{'a', 'c', 'b', (5, 4)}
```

Using the update() function:

```
>> myset.update([1, 2, 3, 4]) # update() only works for iterable objects
>> myset
{'a', 1, 'c', 'b', 4, 2, (5, 4), 3}
>> myset.update({1, 7, 8})
>> myset
{'a', 1, 'c', 'b', 4, 7, 8, 2, (5, 4), 3}
>> myset.update({1, 6}, [5, 13])
>> myset
{'a', 1, 'c', 'b', 4, 5, 6, 7, 8, 2, (5, 4), 13, 3}
```

REMOVING ITEMS

Both the discard() and remove() functions take a single value as an argument and removes that value from the set. If that value is not present, discard() does nothing, but remove() will raise a KeyError exception.

```
>> myset.discard(10)
>> myset
{'a', 1, 'c', 'b', 4, 5, 7, 8, 2, 12, (5, 4), 13, 11, 3}
>> myset.remove(13)
>> myset
{'a', 1, 'c', 'b', 4, 5, 7, 8, 2, 12, (5, 4), 11, 3}
```

COMMON SET OPERATIONS Using union(), intersection() and difference() functions.

```
>> a = {2, 4, 5, 9}
>> b = {2, 4, 11, 12}
>> a.union(b) # Values which exist in a or b
{2, 4, 5, 9, 11, 12}
>> a.intersection(b) # Values which exist in a and b
{2, 4}
>> a.difference(b) # Values which exist in a but not in b
{9, 5}
```

The union() and intersection() functions are symmetric methods:

```
>> a.union(b) == b.union(a)
True
>> a.intersection(b) == b.intersection(a)
True
>> a.difference(b) == b.difference(a)
False
```

These other built-in data structures in Python are also useful.

Task

Given 2 sets of integers, M and N, print their symmetric difference in ascending order. The term symmetric difference indicates those values that exist in either M or N but do not exist in both.

Input Format

The first line of input contains an integer, M.
The second line contains M space-separated integers.
The third line contains an integer, N.
The fourth line contains N space-separated integers.

Output Format

Output the symmetric difference integers in ascending order, one per line.

Sample Input

STDIN Function

---

4 set a size M = 4

2 4 5 9 a = {2, 4, 5, 9}

4 set b size N = 4

2 4 11 12 b = {2, 4, 11, 12}

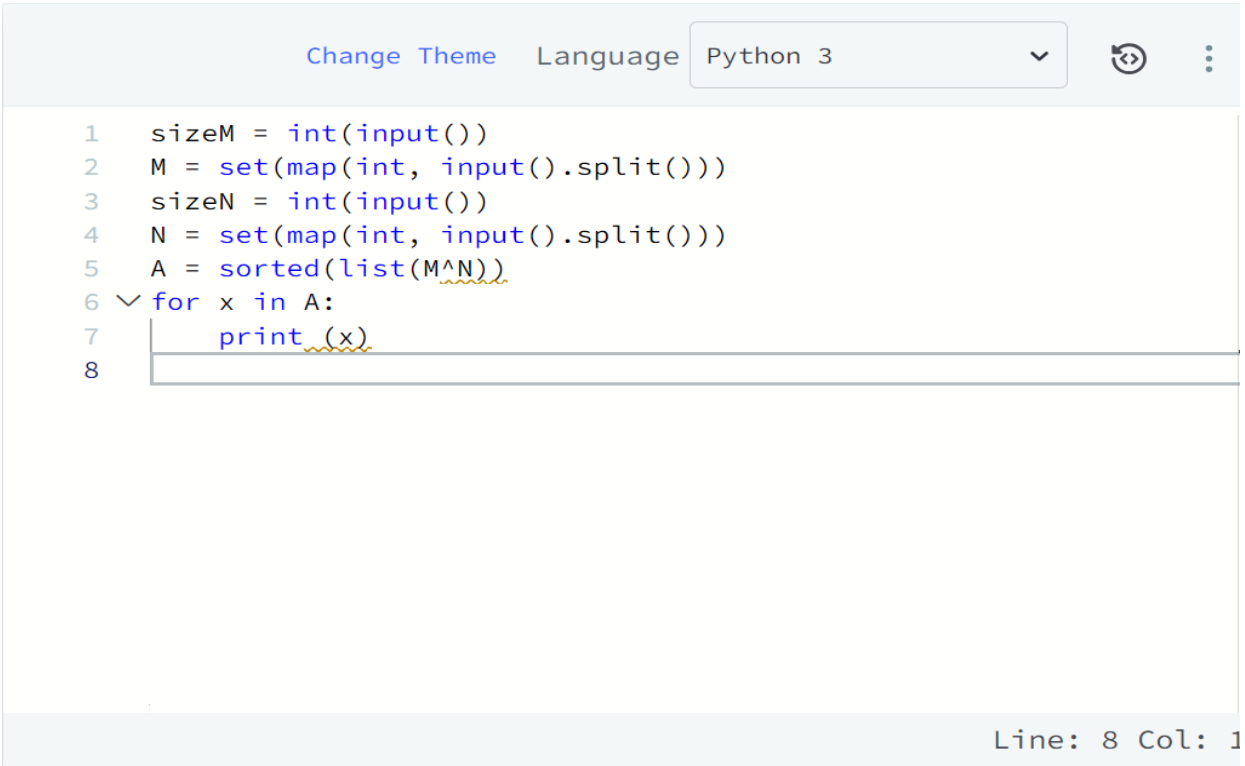
Sample Output

5

9

11

12



Challenge 3:- Set.add() :-

If we want to add a single element to an existing set, we can use the .add() operation. It adds the element to the set and returns 'None'.

Example

```
>>> s = set('HackerRank')
>>> s.add('H')
>>> print s
set(['a', 'c', 'e', 'H', 'k', 'n', 'r', 'R'])
>>> print s.add('HackerRank')
None
>>> print s
set(['a', 'c', 'e', 'HackerRank', 'H', 'k', 'n', 'r', 'R'])
```

Task

Apply your knowledge of the .add() operation to help your friend Rupal.

Rupal has a huge collection of country stamps. She decided to count the total number of distinct country stamps in her collection. She asked for your help. You pick the stamps one by one from a stack of N country stamps.

Find the total number of distinct country stamps.

Input Format

The first line contains an integer N, the total number of country stamps.
The next N lines contains the name of the country where the stamp is from.

Constraints

O<N<1000

Output Format

Output the total number of distinct country stamps on a single line.

Sample Input

```
7
UK
China
USA
France
New Zealand
UK
```

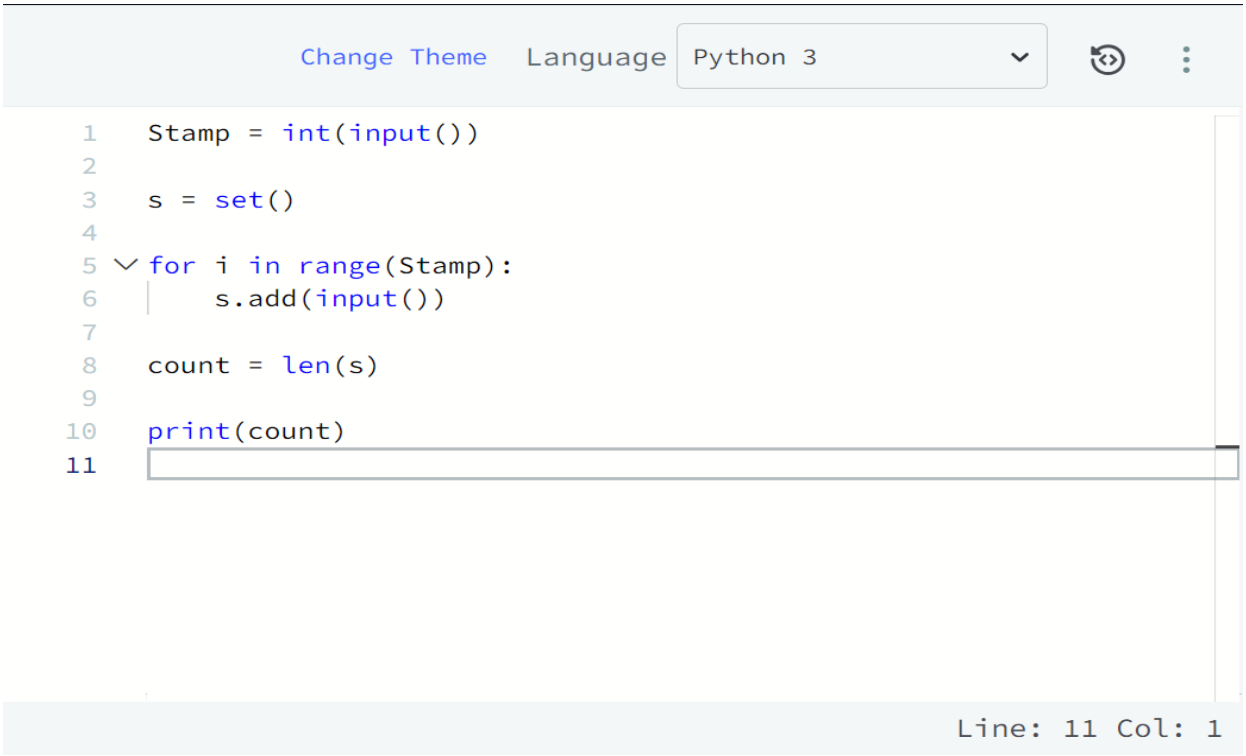
France

Sample Output

5

Explanation

UK and France repeat twice. Hence, the total number of distinct country stamps is 5 (five).



Challenge 4:- Set .discard(), .remove() & .pop() :-

.remove(x)

This operation removes element x from the set.
If element x does not exist, it raises a KeyError.
The .remove(x) operation returns None.

Example

>>> s = set([1, 2, 3, 4, 5, 6, 7, 8, 9])

>>> s.remove(5)

>>> print s

set([1, 2, 3, 4, 6, 7, 8, 9])

>>> print s.remove(4)

None

```
>>> print s
set([1, 2, 3, 6, 7, 8, 9])
>>> s.remove(0)
KeyError: 0
```

.discard(x)

This operation also removes element x from the set.
If element x does not exist, it **does not** raise a KeyError.
The .discard(x) operation returns None.

Example

```
>>> s = set([1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> s.discard(5)
>>> print s
set([1, 2, 3, 4, 6, 7, 8, 9])
>>> print s.discard(4)
None
>>> print s
set([1, 2, 3, 6, 7, 8, 9])
>>> s.discard(0)
>>> print s
set([1, 2, 3, 6, 7, 8, 9])
```

.pop()

This operation removes and return an arbitrary element from the set.
If there are no elements to remove, it raises a KeyError.

Example

```
>>> s = set([1])
>>> print s.pop()
1
>>> print s
set([])
>>> print s.pop()
KeyError: pop from an empty set
```

Task

You have a non-empty set s, and you have to execute N commands given in N lines.

The commands will be pop, remove and discard.

Input Format

The first line contains integer n, the number of elements in the set s .
The second line contains n space separated elements of set s. All of the elements are non-negative integers, less than or equal to 9.
The third line contains integer N, the number of commands.
The next N lines contains either pop, remove and/or discard commands followed by their associated value.

Constraints

0<n<20

0<N<20

Output Format

Print the sum of the elements of set on a single line.

Sample Input

9

1 2 3 4 5 6 7 8 9

10

pop

remove 9

discard 9

discard 8

remove 7

pop

discard 6

remove 5

pop

discard 5

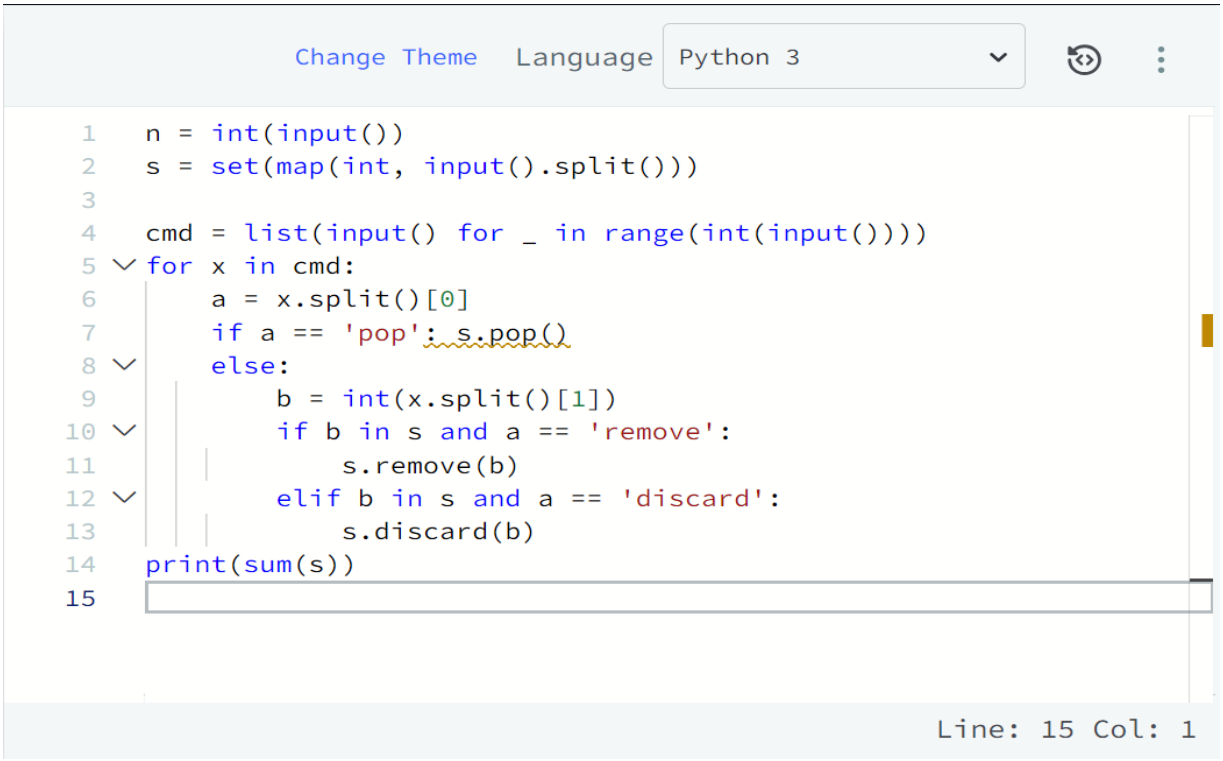
Sample Output

4

Explanation

After completing these 10 operations on the set, we get set ([4]). Hence, the sum is 4.

Note: Convert the elements of set s to integers while you are assigning them. To ensure the proper input of the set, we have added the first two lines of code to the editor.



Challenge 5:- Collections.deque() :-

A deque is a double-ended queue. It can be used to add or remove elements from both ends.

Deques support thread safe, memory efficient appends and pops from either side of the deque with approximately the same O(1) performance in either direction.

Click on the link to learn more about **deque() methods**.
Click on the link to learn more about various approaches to working with deques: **Deque Recipes**.

Examp

le Code

```
>>> from collections import deque
>>> d = deque()
>>> d.append(1)
>>> print d
deque([1])
>>> d.appendleft(2)
>>> print d
deque([2, 1])
>>> d.clear()
```

```
>>> print d
deque([])
>>> d.extend('1')
>>> print d
deque(['1'])
>>> d.extendleft('234')
>>> print d
deque(['4', '3', '2', '1'])
>>> d.count('1')
1
>>> d.pop()
'1'
>>> print d
deque(['4', '3', '2'])
>>> d.popleft()
'4'
>>> print d
deque(['3', '2'])
>>> d.extend('7896')
>>> print d
deque(['3', '2', '7', '8', '9', '6'])
>>> d.remove('2')
>>> print d
deque(['3', '7', '8', '9', '6'])
>>> d.reverse()
>>> print d
deque(['6', '9', '8', '7', '3'])
>>> d.rotate(3)
>>> print d
deque(['8', '7', '3', '6', '9'])
```

Task

Perform append, pop, popleft and appendleft methods on an empty deque d.

Input Format

The first line contains an integer N, the number of operations.
The next N lines contains the space separated names of methods and their values.

Constraints

0<N<=100

Output Format

Print the space separated elements of deque d.

Sample Input

6

append 1

append 2

append 3

appendleft 4

pop

popleft

Sample Output

1 2