

[Dashboard](#) / [My courses](#) / [PSPP/PUP](#) / [Experiments based on Tuples, Sets and its operations](#) / [Week7 Coding](#)

Started on	Friday, 24 May 2024, 8:08 AM
State	Finished
Completed on	Saturday, 25 May 2024, 9:26 AM
Time taken	1 day 1 hour
Marks	5.00/5.00
Grade	100.00 out of 100.00

Question 1

Correct

Mark 1.00 out of 1.00

Write a program to eliminate the common elements in the given 2 arrays and print only the non-repeating elements and the total number of such non-repeating elements.

Input Format:

The first line contains space-separated values, denoting the size of the two arrays in integer format respectively.

The next two lines contain the space-separated integer arrays to be compared.

[Sample](#) Input:

```
5 4
1 2 8 6 5
2 6 8 10
```

[Sample](#) Output:

```
1 5 10
3
```

[Sample](#) Input:

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

[Sample](#) Output:

```
NO SUCH ELEMENTS
```

For example:

Input	Result
5 4 1 2 8 6 5 2 6 8 10	1 5 10 3

Answer: (penalty regime: 0 %)

```
1 | n, m = map(int, input().split())
2 | array1 = list(map(int, input().split()))
3 | array2 = list(map(int, input().split()))
4 | set1 = set(array1)
5 | set2 = set(array2)
6 | symmetric_diff = set1.symmetric_difference(set2)
7 | non_repeating_elements = [x for x in symmetric_diff if x not in set1 or x not in set2]
8 | if non_repeating_elements:
9 |     print(*non_repeating_elements)
10 |    print(len(non_repeating_elements))
11 | else:
12 |    print("NO SUCH ELEMENTS")
```

	Input	Expected	Got	
✓	5 4 1 2 8 6 5 2 6 8 10	1 5 10 3	1 5 10 3	✓
✓	3 3 10 10 10 10 11 12	11 12 2	11 12 2	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.



Question **2**

Correct

Mark 1.00 out of 1.00

Given an array of integers `nums` containing $n + 1$ integers where each integer is in the range $[1, n]$ inclusive. There is only **one repeated number** in `nums`, return *this repeated number*. Solve the problem using [set](#).

Example 1:Input: `nums = [1,3,4,2,2]`

Output: 2

Example 2:Input: `nums = [3,1,3,4,2]`

Output: 3

For example:

Input	Result
1 3 4 4 2	4

Answer: (penalty regime: 0 %)

```

1 def find_duplicate(nums):
2     seen=set()
3     for num in nums:
4         if num in seen:
5             return num
6         seen.add(num)
7 nums=list(map(int, input().split()))
8 print(find_duplicate(nums))
9
10

```

	Input	Expected	Got	
✓	1 3 4 4 2	4	4	✓
✓	1 2 2 3 4 5 6 7	2	2	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question 3

Correct

Mark 1.00 out of 1.00

Given an array of [strings](#) words, return the words that can be typed using letters of the alphabet on only one row of American keyboard like the image below.

In the **American keyboard**:

- the first row consists of the characters "qwertyuiop",
- the second row consists of the characters "asdfghjkl", and
- the third row consists of the characters "zxcvbnm".

~ ,	! 1	@ 2	# 3	\$ 4	% 5	^ 6	& 7	* 8	(9) 0	- _	+ =	Backspace
Tab	Q	W	E	R	T	Y	U	I	O	P	{ [}]	 \ _
Caps Lock	A	S	D	F	G	H	J	K	L	:	" '	Enter	
Shift	Z	X	C	V	B	N	M	< ,	> .	? /	Shift		
Ctrl	Win Key	Alt							Alt	Win Key	Menu	Ctrl	

Example 1:

Input: words = ["Hello", "Alaska", "Dad", "Peace"]

Output: ["Alaska", "Dad"]

Example 2:

Input: words = ["omk"]

Output: []

Example 3:

Input: words = ["adsdf", "sfd"]

Output: ["adsdf", "sfd"]

For example:

Input	Result
4 Hello Alaska Dad Peace	Alaska Dad
2 adsdf afd	adsdf afd

Answer: (penalty regime: 0 %)

```

1 A = int(input())
2 words = [input() for _ in range(A)]
3 rows = [set("qwertyuiop"), set("asdfghjkl"), set("zxcvbnm")]
4 result = [word for word in words if any(set(word.lower()).issubset(row) for row in rows)]
5 if result:

```

```
6 | print("\n".join(result))
7 | else:
8 | print("No words")
```

	Input	Expected	Got	
✓	4 Hello Alaska Dad Peace	Alaska Dad	Alaska Dad	✓
✓	1 omk	No words	No words	✓
✓	2 adsfd afd	adsfd afd	adsfd afd	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question 4

Correct

Mark 1.00 out of 1.00

The **DNA sequence** is composed of a series of nucleotides abbreviated as 'A', 'C', 'G', and 'T'.

- For example, "ACGAATTCG" is a **DNA sequence**.

When studying **DNA**, it is useful to identify repeated sequences within the DNA.

Given a string `s` that represents a **DNA sequence**, return all the **10-letter-long** sequences (substrings) that occur more than once in a DNA molecule. You may return the answer in **any order**.

Example 1:

Input: `s = "AAAAACCCCCAAAAACCCCCAAAAGGGTTT"`

Output: `["AAAAACCCCC", "CCCCAAAAA"]`

Example 2:

Input: `s = "AAAAAAAAAAAA"`

Output: `["AAAAAAAAA"]`

For example:

Input	Result
AAAAACCCCCAAAAACCCCCAAAAGGGTTT	AAAAACCCCC CCCCAAAAA

Answer: (penalty regime: 0 %)

```

1 | s = input()
2 | A = set()
3 | B = set()
4 | for i in range(len(s) - 9):
5 |     C = s[i:i + 10]
6 |     if C in A:
7 |         B.add(C)
8 |     else:
9 |         A.add(C)
10 | for seq in B:
11 |     print(seq)
```

	Input	Expected	Got	
✓	AAAAACCCCCAAAAACCCCCAAAAGGGTTT	AAAAACCCCC CCCCAAAAA	AAAAACCCCC CCCCAAAAA	✓

	Input	Expected	Got	
✓	AAAAAAAAAAAAA	AAAAAAAAAA	AAAAAAAAAA	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **5**

Correct

Mark 1.00 out of 1.00

Coders here is a simple task for you, Given string str. Your task is to check whether it is a binary string or not by using python [set](#).

Examples:

Input: str = "01010101010"

Output: Yes

Input: str = "REC101"

Output: No

For example:

Input	Result
01010101010	Yes
010101 10101	No

Answer: (penalty regime: 0 %)

```

1 def is_binary_string(s):
2     binary_set = {'0', '1'}
3     return set(s).issubset(binary_set)
4
5 def main():
6     s = input().strip()
7     if is_binary_string(s):
8         print("Yes")
9     else:
10        print("No")
11
12 if __name__ == "__main__":
13     main()
14

```

	Input	Expected	Got	
✓	01010101010	Yes	Yes	✓
✓	REC123	No	No	✓
✓	010101 10101	No	No	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

[◀ Week7_MCQ](#)

Jump to...

[Dictionary ▶](#)