SPEEDRUN THE PYTHON CODING INTERVIEW:

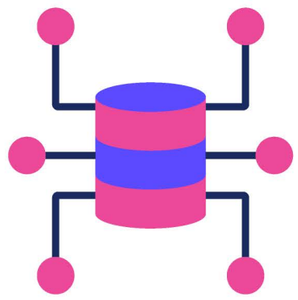
WHAT TO EXPECT FROM APPLE

educative

RELEVANT CODING PATTERNS TO REVIEW

- **Custom Data Structures**
- **Knowing What to Track**
- **Sliding Window**
- **Two Pointers**
- **Dynamic Programming**
- **Stacks**
- **Top K Elements**
- **K-Way Merge**

CUSTOM DATA STRUCTURES EXAMPLE PROBLEM:



Implement Least Recently Used (LRU) cache:

- `Init(capacity):` Initializes an LRU cache with the capacity size.
- `Set(key, value):` Adds a new key-value pair or updates an existing key.
- `Get(key):` Returns the value of the key, or −1 if the key does not exist.

If the number of keys has reached the cache capacity, evict the least recently used key and add the new key.

Input:

```
key = 2
```

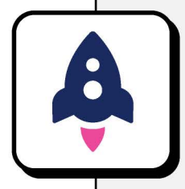
```
Get(2)
cache size = 4
```

key	value	
4	4	← LRU
5	5	
3	3	
2	2	

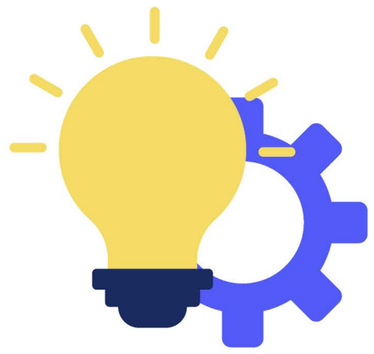
We are getting 2 here, so 2 is updated as the most recent used key-value pair and is no longer candidate for eviction, but 4 is.

Output:

```
2
```



KNOWING WHAT TO TRACK EXAMPLE PROBLEM:



Two Sum:

- For the given array of integers `arr` and a target `t`, you have to identify the two indices that add up to generate the target `t`. Moreover, you can't use the same index twice, and there will be only one solution.

***Note:** We will assume that the array is zero-indexed and the output order doesn't matter.

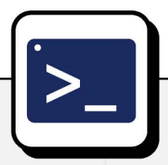
Input:

arr	-2	2	11	15	-3

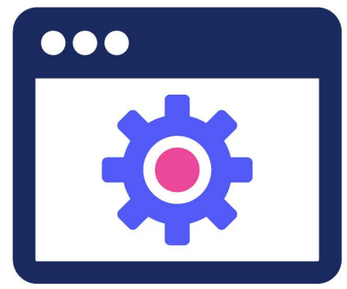
target	8

Output:

```
[2, 4]
```



SLIDING WINDOW EXAMPLE PROBLEM:



Longest Substring without Repeating Characters:

- Given a string, `input_str`, return the length of the longest substring without repeating characters.

Constraints:

- $1 \leq input\_str.length \leq 10^5$
- `input_str` consists of English letters, digits, and spaces.

Input:

string	p	w	w	k	e

Output:

```
length = 3
```

TWO POINTERS EXAMPLE PROBLEM:

Trapping Rain Water:

- Given a sequence of non-negative integers representing the heights of bars in an elevation map, the goal is to determine the amount of rainwater that can be trapped between the bars.

Constraints:

- `n== heights.length`
- $0 \leq$ `heights[i]` $\leq 10^5$
- $1 \leq n \leq 10^3$

Input:

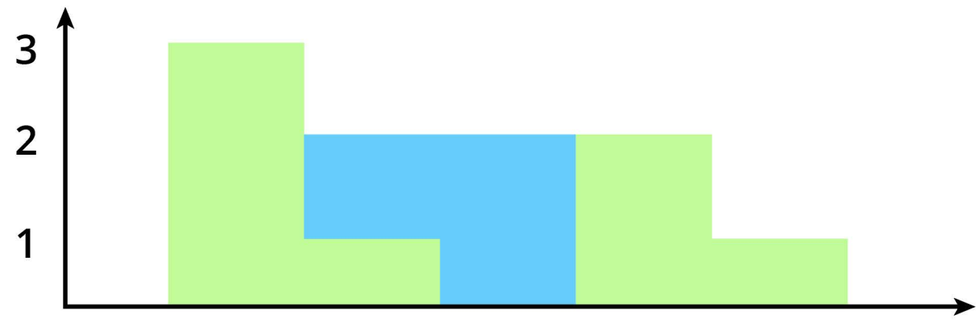
heights	3	1	0	2	1

Output:

output	3

Explanation

There are 3 units of rain water (blue bars) trapped in the following given elevation map (green bars)



DYNAMIC PROGRAMMING EXAMPLE PROBLEM:

Longest Palindromic Substring:

- Given a string `s`, return the longest palindromic substring in `s`.

Constraints:

- `1≤s.length ≤1000`
- `s` consist of only digits and English letters.

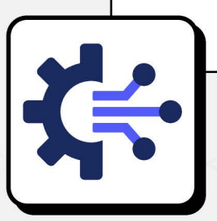
Input:

s	"asfdodfsaiuoefbwjebejwbf"

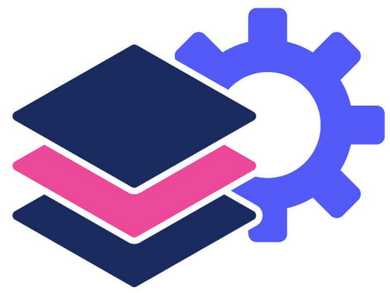
Output:

s	"fbwjebejwbf"

Although there are several palindromic substrings in the input string, we only return the longest occurring palindrome.



STACKS
EXAMPLE PROBLEM:



Valid Parentheses:

- Given a string that may consist of opening and closing parentheses, your task is to check whether or not the string contains valid parenthesization.

Constraints:

- Every opening parenthesis should be closed by the same kind of parenthesis. Therefore, `{)` and `[(` strings are invalid.
- Every opening parenthesis must be closed in the correct order. Therefore, `)(` and `()(()` are invalid.

Input:

```
(){[{()}]}
```

Output:

```
TRUE
```

TOP K ELEMENTS EXAMPLE PROBLEM:

Top K Frequent Elements:

- Given an array of integers, arr, and an integer, k, return the `k`k most frequent elements.

***Note:** You can return the answer in any order.

Constraints:

- `1 ≤arr.length ≤10³`
- `10⁻⁴≤ arr[i] ≤10⁴`
- `1 ≤k ≤`number of unique elements in an array.

Input:

k	2

arr	1	3	5	14	18	14	5

Output:

result	5	14

K-WAY MERGE EXAMPLE PROBLEM:

Merge Sorted Array:

- Given two sorted integer arrays, `nums1` and `nums2`, and the number of data elements in each array, `m` and `n`, implement a function that merges the second array into the first one. You have to modify `nums1` in place.

***Note:** Assume that `nums1` has a size equal to `m+n`, meaning it has enough space to hold additional elements from `nums2`.

Input:

nums1	1	4	9	0	0

m=3

nums2	1	76

n=2

Output:

1	1	4	9	76