```
{ { "id": 3, "title": "Longest Substring Without Repeating Characters", "testcases":
[ { "input": { "s": "abcabcbb" }, "output": 3 },

{ "input": { "s": "" }, "output": 0 },
{ "input": { "s": "a" }, "output": 1 },
{ "input": { "s": "aa" }, "output": 1 },
{ "input": { "s": "ab" }, "output": 2 },
{ "input": { "s": "abac" }, "output": 3 },
{ "input": { "s": "abcdefghijklmnopqrstuvwxyz" }, "output": 26 },
{ "input": { "s": "aaaaaaaaaaaaaaaaaaaaaaaaa" }, "output": 1 },
{ "input": { "s": "abcdabcdabcdabcdabcdabcd" }, "output": 4 },
{ "input": { "s": "abcdeabcdeabcdeabcdeabcde" }, "output": 5 },
{ "input": { "s": "aaaaaaaaaa" }, "output": 1 },
{ "input": { "s": "abcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabc" }, "output":
3 },
{ "input": { "s": "a".repeat(10000) }, "output": 1 },
{ "input": { "s": "abcdefg".repeat(700) }, "output": 7 },
{ "input": { "s": " ".repeat(10000) }, "output": 1 },
{ "input": { "s": "0123456789".repeat(500) }, "output": 10 },
{ "input": { "s": "AaBbCcDdEeFfGgHhIiJj" }, "output": 20 },
{ "input": { "s": "abcdefghijklmnopqrstuvwxyza" }, "output": 26 },
{ "input": { "s": "z".repeat(5000) + "a" + "z".repeat(4999) }, "output": 2 },
{ "input": { "s": "a".repeat(4999) + "b" }, "output": 2 },
{ "input": { "s": "".join(chr(65+i%26) for i in range(50000)) }, "output": 26 },

{ "input": { "s": "a1b2c3d4e5f6g7" }, "output": 14 },
{ "input": { "s": "abcABC123!@#" }, "output": 12 },
{ "input": { "s": "abccba" }, "output": 3 },
{ "input": { "s": "1234567890" }, "output": 10 },
{ "input": { "s": "!@#$%^&*()" }, "output": 10 },
{ "input": { "s": "thequickbrownfoxjumpsoverthelazydog" }, "output": 26 },
{ "input": { "s": "AaBbCcDdEeFf" }, "output": 12 },
{ "input": { "s": "abc def ghi" }, "output": 7 },
{ "input": { "s": "a b c d e f g" }, "output": 13 },
{ "input": { "s": "  a  " }, "output": 2 },
{ "input": { "s": "____" }, "output": 1 },
{ "input": { "s": "abcabcabcabcabcabc" }, "output": 3 },
{ "input": { "s": "a1!a2@b3#c4$d5%" }, "output": 15 },
{ "input": { "s": "a".repeat(100) + "b" }, "output": 2 },
{ "input": { "s": "ab" + "a".repeat(9998) }, "output": 2 },

{ "input": { "s": "aAaAaAaA" }, "output": 2 },
{ "input": { "s": "aA bB cC" }, "output": 7 },
{ "input": { "s": "😀😃😄😁😆😅😂🤣😊😇" }, "output": 10 },
{ "input": { "s": "a\nb\tc\rd" }, "output": 5 },
{ "input": { "s": "abcdedcba" }, "output": 5 },
{ "input": { "s": "1122334455" }, "output": 2 },
{ "input": { "s": "abcde" + "a" }, "output": 5 },
{ "input": { "s": "aaaabbbbccccddddeeeeffffgggg" }, "output": 2 },
```

{ "input": { "s": "a" + "b".repeat(4999) }, "output": 2 },
{ "input": { "s": "a".repeat(4999) + "c" }, "output": 2 },
{ "input": { "s": "ab".repeat(25000) }, "output": 2 },
{ "input": { "s": "".join(chr(33 + i % 94) for i in range(50000)) }, "output": 94 },
{ "input": { "s": "s" }, "output": 1 },
{ "input": { "s": "aBcDeFgHiJkLmNoPqRsTuVwXyZ" }, "output": 26 },
{ "input": { "s": "ABCDabcdABCDabcd" }, "output": 8 },

{ "input": { "s": "hello" }, "output": 3 },
{ "input": { "s": "world" }, "output": 5 },
{ "input": { "s": "mississippi" }, "output": 3 },
{ "input": { "s": "banana" }, "output": 2 },
{ "input": { "s": "google" }, "output": 4 },
{ "input": { "s": "facebook" }, "output": 6 },
{ "input": { "s": "linkedin" }, "output": 6 },
{ "input": { "s": "twitter" }, "output": 5 },
{ "input": { "s": "amazon" }, "output": 4 },
{ "input": { "s": "123abc456def" }, "output": 12 },
{ "input": { "s": "xyzxyzxyz" }, "output": 3 },
{ "input": { "s": "aababcabcd" }, "output": 4 },
{ "input": { "s": "cccccccccccccccc" }, "output": 1 },
{ "input": { "s": "pythonrocks" }, "output": 10 },
{ "input": { "s": "openai" }, "output": 6 },
{ "input": { "s": "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ" }, "output": 52 },
{ "input": { "s": "abcdeffedcba" }, "output": 6 },
{ "input": { "s": "abc def ghi jkl mno" }, "output": 7 },
{ "input": { "s": "a!b@c#d$e%f^g&h*i(j)k" }, "output": 21 },
{ "input": { "s": "zxywvutsrqponmlkjihgfedcba" }, "output": 26 },
{ "input": { "s": "abacabadabacabae" }, "output": 5 },
{ "input": { "s": "ab".repeat(1000) }, "output": 2 },
{ "input": { "s": "a1a2a3a4a5a6a7a8" }, "output": 3 },
{ "input": { "s": "12345abcde54321" }, "output": 10 },
{ "input": { "s": "abc123!@#ABC" }, "output": 12 }

] }, { "id": 2, "title": "Add Two Numbers", "testcases": [{ "input": { "l1": [0], "l2":
[0] }, "output": [0] },

{ "input": { "l1": [1], "l2": [9] }, "output": [0,1] },
{ "input": { "l1": [2], "l2": [3] }, "output": [5] },
{ "input": { "l1": [9], "l2": [9] }, "output": [8,1] },
{ "input": { "l1": [5,6], "l2": [5,4] }, "output": [0,1,1] },
{ "input": { "l1": [2,4,3], "l2": [5,6,4] }, "output": [7,0,8] },
{ "input": { "l1": [0,1], "l2": [0,9] }, "output": [0,0,1] },
{ "input": { "l1": [1,0,0], "l2": [9,9] }, "output": [0,0,1] },
{ "input": { "l1": [9,9,9], "l2": [1] }, "output": [0,0,0,1] },
{ "input": { "l1": [1,2,3], "l2": [0,0,0] }, "output": [1,2,3] },
{ "input": { "l1": [0,0,1], "l2": [0,0,1] }, "output": [0,0,2] },
{ "input": { "l1": [9]*100, "l2": [1] }, "output": [0]*100 + [1] },
{ "input": { "l1": [1]*100, "l2": [2]*100 }, "output": [3]*100 },

```
{ "input": { "l1": [1,2,3], "l2": [7,7,6] }, "output": [8,9,9] },
{ "input": { "l1": [0]*99 + [1], "l2": [1] }, "output": [1]*99 + [1] },
{ "input": { "l1": [1], "l2": [0]*99 + [1] }, "output": [1]*99 + [2] },
{ "input": { "l1": [9]*50, "l2": [9]*50 }, "output": [8]+[9]*49+[0]+[1] },
{ "input": { "l1": [1,0,0,0,0,0,0], "l2": [9,9,9,9,9,9] }, "output": [0,0,0,0,0,0,1] },
{ "input": { "l1": [0], "l2": [1]*100 }, "output": [1]*100 },
{ "input": { "l1": [1]*100, "l2": [0] }, "output": [1]*100 },

{ "input": { "l1": [1,2,3,4,5], "l2": [5,4,3,2,1] }, "output": [6,6,6,6,6] },
{ "input": { "l1": [9,8,7], "l2": [1,2,3] }, "output": [0,1,1,1] },
{ "input": { "l1": [9,9], "l2": [2] }, "output": [1,0,1] },
{ "input": { "l1": [1,2,3,4], "l2": [6,7,8] }, "output": [7,9,1,5] },
{ "input": { "l1": [0,0,0,1], "l2": [9,9,9] }, "output": [9,9,9,1] },
{ "input": { "l1": [5,5,5,5], "l2": [5,5,5,5] }, "output": [0,1,1,1,1] },
{ "input": { "l1": [0,0,1], "l2": [1] }, "output": [1,0,1] },
{ "input": { "l1": [0]*100, "l2": [1]*100 }, "output": [1]*100 },
{ "input": { "l1": [1]*100, "l2": [9]*100 }, "output": [0]*100 + [1] },
{ "input": { "l1": [9]*99 + [1], "l2": [1] }, "output": [0]*99 + [2] },

{ "input": { "l1": [3], "l2": [7] }, "output": [0,1] },
{ "input": { "l1": [5], "l2": [5] }, "output": [0,1] },
{ "input": { "l1": [2,4,3], "l2": [5,6,4] }, "output": [7,0,8] },
{ "input": { "l1": [9,9,9], "l2": [9,9,9] }, "output": [8,9,9,1] },
{ "input": { "l1": [1,0,0,0,0], "l2": [9,9,9,9] }, "output": [0,0,0,0,1] },
{ "input": { "l1": [1], "l2": [9,9,9,9,9,9,9,9] }, "output": [0,0,0,0,0,0,0,0,1] },
{ "input": { "l1": [1,2], "l2": [9] }, "output": [0,3] },
{ "input": { "l1": [2], "l2": [8] }, "output": [0,1] },
{ "input": { "l1": [0,1,2], "l2": [9,8] }, "output": [9,9,2] },
{ "input": { "l1": [1,1,1], "l2": [8,8,8] }, "output": [9,9,9] },

{ "input": { "l1": [1,8], "l2": [0] }, "output": [1,8] },
{ "input": { "l1": [5,6,7,8,9], "l2": [5,6,7,8,9] }, "output": [0,3,5,7,9,1] },
{ "input": { "l1": [9,9,9,9,9], "l2": [1] }, "output": [0,0,0,0,0,1] },
{ "input": { "l1": [0], "l2": [9,9,9,9,9] }, "output": [9,9,9,9,9] },
{ "input": { "l1": [1,1,1,1], "l2": [9,9,9,9] }, "output": [0,1,1,1,1] },
{ "input": { "l1": [0,0,0], "l2": [1,2,3] }, "output": [1,2,3] },
{ "input": { "l1": [1,0,1], "l2": [1,0,1] }, "output": [2,0,2] },
{ "input": { "l1": [3,4], "l2": [7,6] }, "output": [0,1,1] },
{ "input": { "l1": [9], "l2": [9,9,9,9] }, "output": [8,0,0,0,1] },
{ "input": { "l1": [2,3], "l2": [7] }, "output": [9,3] },

{ "input": { "l1": [1,2,3,4,5,6,7,8,9], "l2": [9,8,7,6,5,4,3,2,1] }, "output":
[0,1,1,1,1,1,1,1,1,1] },
{ "input": { "l1": [1]*50, "l2": [9]*50 }, "output": [0]*50 + [1] },
{ "input": { "l1": [4,5,6], "l2": [5,4,3] }, "output": [9,9,9] },
{ "input": { "l1": [2,4,6,8], "l2": [1,3,5,7] }, "output": [3,7,1,6,1] },
{ "input": { "l1": [0,0,0,0,1], "l2": [1] }, "output": [1,0,0,0,1] },
{ "input": { "l1": [9]*99, "l2": [1] }, "output": [0]*99 + [1] },
{ "input": { "l1": [3], "l2": [9,9,9] }, "output": [2,0,0,1] },
```

```
{ "input": { "l1": [0], "l2": [9,9,9] }, "output": [9,9,9] },
{ "input": { "l1": [9], "l2": [1,0,0] }, "output": [0,1,0] },
{ "input": { "l1": [5,5,5], "l2": [5,5,5] }, "output": [0,1,1,1] }


] }, { "id": 5, "title": "Longest Palindromic Substring", "testcases": [ { "input": { "s":
"babad" }, "output": "bab" },

{ "input": { "s": "a" }, "output": "a" },
{ "input": { "s": "aa" }, "output": "aa" },
{ "input": { "s": "abc" }, "output": "a" },
{ "input": { "s": "aba" }, "output": "aba" },
{ "input": { "s": "abba" }, "output": "abba" },
{ "input": { "s": "abcdedcba" }, "output": "abcdedcba" },
{ "input": { "s": "abcdefg" }, "output": "a" },
{ "input": { "s": "abcba" }, "output": "abcba" },
{ "input": { "s": "racecar" }, "output": "racecar" },
{ "input": { "s": "a"*1000 }, "output": "a"*1000 },
{ "input": { "s": "abacdfgdcaba" }, "output": "aba" },
{ "input": { "s": "abcddcba" }, "output": "abcddcba" },
{ "input": { "s": "forgeeksskeegfor" }, "output": "geeksskeeg" },
{ "input": { "s": "a"*999 + "b" }, "output": "a"*999 },
{ "input": { "s": "b" + "a"*999 }, "output": "a"*999 },
{ "input": { "s": "abc" * 300 + "a" }, "output": "a" },
{ "input": { "s": "abba"*250 }, "output": "abba" },
{ "input": { "s": "abcd"*250 }, "output": "a" },
{ "input": { "s": "a"*499 + "b" + "a"*500 }, "output": "a"*499 + "b" + "a"*500 },
{ "input": { "s": "a"*500 + "b" + "a"*499 }, "output": "a"*500 + "b" + "a"*499 },

{ "input": { "s": "cbbd" }, "output": "bb" },
{ "input": { "s": "abcda" }, "output": "a" },
{ "input": { "s": "abcbaaa" }, "output": "abcba" },
{ "input": { "s": "aaaabcaaa" }, "output": "aaa" },
{ "input": { "s": "abcdxyzyxabcdaaa" }, "output": "xyzyx" },
{ "input": { "s": "a"*10 }, "output": "a"*10 },
{ "input": { "s": "abcdedcbazz" }, "output": "abcdedcba" },
{ "input": { "s": "abc12321cba" }, "output": "abc12321cba" },
{ "input": { "s": "abc1234321cba" }, "output": "abc1234321cba" },
{ "input": { "s": "abcdefgfedcba" }, "output": "abcdefgfedcba" },

{ "input": { "s": "abcxyzzyxabc" }, "output": "xyzzyx" },
{ "input": { "s": "a"*500 + "xyz" + "a"*497 }, "output": "a"*500 },
{ "input": { "s": "abcdcbaefg" }, "output": "abcdcba" },
{ "input": { "s": "aaabaaaa" }, "output": "aaabaaa" },
{ "input": { "s": "abcbabcbabcba" }, "output": "abcbabcbabcba" },
{ "input": { "s": "abbaabba" }, "output": "abbaabba" },
{ "input": { "s": "abaaba" }, "output": "abaaba" },
{ "input": { "s": "a"*250 + "b"*250 + "a"*250 }, "output": "a"*250 },
{ "input": { "s": "xyz" + "a"*300 + "xyz" }, "output": "a"*300 },
{ "input": { "s": "abcdef" }, "output": "a" },
```

```
{ "input": { "s": "aaaaabaaaaa" }, "output": "aaaaabaaaaa" },
{ "input": { "s": "aabbaa" }, "output": "aabbaa" },
{ "input": { "s": "xyzabcdedcba123" }, "output": "abcdedcba" },
{ "input": { "s": "abaxyzzyxf" }, "output": "xyzzyx" },
{ "input": { "s": "abacdfgdcaba" }, "output": "aba" },
{ "input": { "s": "abcdaabbaa" }, "output": "aabbaa" },
{ "input": { "s": "abcbadeffed" }, "output": "deffed" },
{ "input": { "s": "banana" }, "output": "anana" },
{ "input": { "s": "civic" }, "output": "civic" },
{ "input": { "s": "noon" }, "output": "noon" },

{ "input": { "s": "level" }, "output": "level" },
{ "input": { "s": "refer" }, "output": "refer" },
{ "input": { "s": "rotator" }, "output": "rotator" },
{ "input": { "s": "madamimadam" }, "output": "madamimadam" },
{ "input": { "s": "redder" }, "output": "redder" },
{ "input": { "s": "abccba" }, "output": "abccba" },
{ "input": { "s": "aabbccddeeffgghhiijj" }, "output": "aa" },
{ "input": { "s": "abcdzdcab" }, "output": "cdzdc" },
{ "input": { "s": "abcbaabcbadef" }, "output": "abcbaabcba" },
{ "input": { "s": "xyz" }, "output": "x" },

{ "input": { "s": "aaaa" }, "output": "aaaa" },
{ "input": { "s": "abacdfgdcabba" }, "output": "abba" },
{ "input": { "s": "abcddcba" }, "output": "abcddcba" },
{ "input": { "s": "abccba" }, "output": "abccba" },
{ "input": { "s": "z" }, "output": "z" },
{ "input": { "s": "zyx" }, "output": "z" },
{ "input": { "s": "xyzzyx" }, "output": "xyzzyx" },
{ "input": { "s": "a"*250 + "xyzzyx" + "a"*250 }, "output": "xyzzyx" },
{ "input": { "s": "abc" * 200 }, "output": "a" },
{ "input": { "s": "abacdgfdcaba" }, "output": "aba" },

{ "input": { "s": "abcdxyzyxabcdaaa" }, "output": "xyzyx" },
{ "input": { "s": "abc123321cba" }, "output": "abc123321cba" },
{ "input": { "s": "a"*999 + "z" }, "output": "a"*999 },
{ "input": { "s": "a" + "b"*998 + "a" }, "output": "b"*998 },
{ "input": { "s": "abcdefgfedcba" }, "output": "abcdefgfedcba" },
{ "input": { "s": "abcdgfedcba" }, "output": "gfedcg" },
{ "input": { "s": "abacdfgdcabba" }, "output": "abba" },
{ "input": { "s": "babaddtattarrattatddetartrateedredividerb" }, "output": "tattarrattat" },
{ "input": { "s": "cbbdxyzzyxabc" }, "output": "xyzzyx" },
{ "input": { "s": "abcdefghijklmnopqrstuvwxyz" }, "output": "a" }


] }, { "id": 15, "title": "3Sum", "testcases": [ { "input": [-1, 0, 1, 2, -1, -4], "output":
[[-1, -1, 2], [-1, 0, 1]] }, { "input": [0, 0, 0], "output": [[0, 0, 0]] }, { "input": [1, 2,
-3], "output": [[-3, 1, 2]] }, { "input": [-1, -1, 2], "output": [[-1, -1, 2]] }, { "input":
```

[1, 1, -2], "output": [[-2, 1, 1]] }, { "input": [0, 1, -1], "output": [[-1, 0, 1]] }, { "input": [100000, -100000, 0], "output": [[-100000, 0, 100000]] }, { "input": [1, 2, 3], "output": [] }, { "input": [3000, 3000, 3000], "output": [] }, { "input": Array(1500).fill(-1).concat(Array(1500).fill(1)), "output": [[-1, -1, 2], [-1, 0, 1]] }, { "input": Array.from({length: 3000}, (_, i) => i - 1500), "output": [] }, { "input": [105, -105, 0], "output": [[-105, 0, 105]] }, { "input": [105, 105, -210], "output": [[-210, 105, 105]] }, { "input": [0, 0, 0, 0, 0], "output": [[0, 0, 0]] }, { "input": Array(100).fill(105).concat(Array(100).fill(-210)).concat([0]), "output": [[-210, 105, 105]] }, { "input": [-105, -105, 210], "output": [[-105, -105, 210]] }, { "input": [105, 104, -209], "output": [[-209, 104, 105]] }, { "input": [-105, 0, 105, 1, -1], "output": [[-105, 0, 105], [-1, 0, 1]] }, { "input": [0, 1, 2, 3, 4], "output": [] }, { "input": [-1, -1, -1, 2, 2, 2], "output": [[-1, -1, 2]] }, { "input": [3, -2, 1, -1, 2, -3], "output": [[-3, 1, 2], [-2, -1, 3]] }, { "input": [0, 1, -1, 0, 0, 0], "output": [[-1, 0, 1], [0, 0, 0]] }, { "input": [], "output": [] }, { "input": [0], "output": [] }, { "input": [0, 0], "output": [] },

// 25 randomized input-only test cases
{
  "input": [-21, 45, 99, 53, 76, -58, -40, -44, 40, -42, -8, -60, -68, 54, 59, 85, 99, -87, 96, -27, -70, 50, -71, 63, 93, -13, 75, -44, -49, -3, -59, 3, -12, 98, -93, -45, -83, -13, -80, 37, -62, 7, -14, 55, 16, -84, 81, -93, 98, -1, -27, 42, -78, -33, 33, -1, -89, -41, -60, 97, -45, -88, -40, -8, 92, -68, 23, -67, 28, 44, 17, 84, 25, -63, 40, 83, -63, 13, -79, -71, -26, -92, -26, 61, -89, -37, -6, -7, -18, 96, -14, 88, -2, -50, -67, -67, 22, -65, 6, -22],
  "output": null
},
{
  "input": [99, 100, -91, 43, 39, 13, 30, -81, 59, 55, 63, -87, 6, -24, 47, 8, 3, -76, -47, 80, -73, -10, -85, -64, -33, 65, -9, -24, -3, -54, 84, 62, 97, 20, -60, 2, -46, 71, -83, 60, -98, 94, 3, -69, -52, 87, -62, 85, -84, -56, -90, -41, 84, 14, -59, -11, -24, -98, -24, 62, 95, 93, 85, -79, 95, -27, -39, 3, 79, 65, 79, 84, -3, -65, -17, 26, -59, -56, -63, -2, -89, 49, -21, 67, -98, -21, -73, -97, 49, -74, -30, 73, 20, -85, 18, -81, -37, -85, 86, -37],
  "output": null
},
{
  "input": [-100, 100, 0],
  "output": [[-100, 0, 100]]
},
{
  "input": [-1, -2, -3, -4, -5],
  "output": []
},
{
  "input": [1, 2, 3, 4, 5],
  "output": []
},
{
  "input": [1, -1, -1, 2, 0, -2],
  "output": [[-2, 0, 2], [-1, -1, 2], [-1, 0, 1]]
},
{
  "input": [4, -2, -2, 0, 1, -1],

```
  "output": [[-2, -1, 3], [-2, 0, 2], [-1, 0, 1]]
},
{
  "input": [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5],
  "output": [[-5, 0, 5], [-5, 1, 4], [-5, 2, 3], [-4, -1, 5], [-4, 0, 4], [-4, 1, 3], [-4, 2,
2], [-3, -2, 5], [-3, -1, 4], [-3, 0, 3], [-3, 1, 2], [-2, -1, 3], [-2, 0, 2], [-2, 1, 1], [-
1, 0, 1]]
},
{
  "input": [0, 0, 1, -1],
  "output": [[-1, 0, 1]]
},
{
  "input": [-10, -7, -3, 0, 2, 4, 6, 7],
  "output": [[-10, 4, 6], [-7, 0, 7], [-3, -1, 4], [-3, 0, 3]]
}
```

] }, { "id": 146, "title": "Container With Most Water", "testcases": [{ "input":
{ "commands": ["LRUCache", "put", "put", "get", "put", "get", "put", "get", "get", "get"],
"args": [[2], [1, 1], [2, 2], [1], [3, 3], [2], [4, 4], [1], [3], [4]] }, "output": [null,
null, null, 1, null, -1, null, -1, 3, 4] },

```
{
  "input": {
    "commands": ["LRUCache", "put", "get", "put", "get", "get"],
    "args": [[1], [1, 10], [1], [2, 20], [1], [2]]
  },
  "output": [null, null, 10, null, -1, 20]
},

{
  "input": {
    "commands": ["LRUCache", "put", "put", "put", "get", "get"],
    "args": [[2], [2, 1], [2, 2], [1, 1], [2], [1]]
  },
  "output": [null, null, null, null, 2, 1]
},

{
  "input": {
    "commands": ["LRUCache", "put", "get"],
    "args": [[1], [10, 100], [10]]
  },
  "output": [null, null, 100]
},

{
  "input": {
    "commands": ["LRUCache", "put", "put", "get", "put", "get", "put", "put", "get", "get"],
```

```
    "args": [[2], [2, 1], [1, 1], [2], [4, 1], [1], [3, 1], [5, 1], [3], [5]]
  },
  "output": [null, null, null, 1, null, 1, null, null, -1, 1]
},

{
  "input": {
    "commands": ["LRUCache", "put", "put", "put", "get", "get", "put", "get", "get"],
    "args": [[2], [2, 6], [1, 5], [1, 2], [1], [2], [4, 7], [1], [4]]
  },
  "output": [null, null, null, null, 2, 6, null, 2, 7]
},

{
  "input": {
    "commands": ["LRUCache", "get"],
    "args": [[1], [5]]
  },
  "output": [null, -1]
},

{
  "input": {
    "commands": ["LRUCache", "put", "put", "put", "put", "get"],
    "args": [[2], [1, 1], [2, 2], [3, 3], [4, 4], [2]]
  },
  "output": [null, null, null, null, null, -1]
},

{
  "input": {
    "commands": ["LRUCache", "put", "get", "put", "get", "get", "put", "put", "put", "put"],
    "args": [[3], [1, 1], [1], [2, 2], [1], [2], [3, 3], [4, 4], [5, 5], [6, 6]]
  },
  "output": [null, null, 1, null, 1, 2, null, null, null, null]
},

{
  "input": {
    "commands": ["LRUCache", "put", "get", "put", "put", "get", "get"],
    "args": [[2], [2, 1], [2], [3, 2], [4, 3], [2], [3]]
  },
  "output": [null, null, 1, null, null, -1, 2]
},

// Size-boundary (shorter examples and max capacity near edge)
{
  "input": {
    "commands": ["LRUCache"] + Array.from({length: 3000}, (_, i) => "put"),
```

```
    "args": [[3000], ...Array.from({length: 3000}, (_, i) => [i, i * 10])]
  },
  "output": [null, ...Array(3000).fill(null)]
},

{
  "input": {
    "commands": ["LRUCache", "put", "put", "get"],
    "args": [[3000], [10000, 1], [9999, 2], [9999]]
  },
  "output": [null, null, null, 2]
},

// Value-boundary
{
  "input": {
    "commands": ["LRUCache", "put", "put", "get", "put", "get"],
    "args": [[2], [10000, 100000], [0, 0], [10000], [1, 1], [0]]
  },
  "output": [null, null, null, 100000, null, -1]
},

{
  "input": {
    "commands": ["LRUCache", "put", "put", "put", "put", "get"],
    "args": [[2], [1, 1], [2, 2], [3, 3], [4, 4], [1]]
  },
  "output": [null, null, null, null, null, -1]
},

// Functional edge/corner
{
  "input": {
    "commands": ["LRUCache", "put", "put", "put", "get", "get", "get"],
    "args": [[3], [1, 10], [2, 20], [3, 30], [2], [1], [3]]
  },
  "output": [null, null, null, null, 20, 10, 30]
},

{
  "input": {
    "commands": ["LRUCache", "put", "get", "put", "get"],
    "args": [[1], [1, 1], [1], [2, 2], [1]]
  },
  "output": [null, null, 1, null, -1]
},

// Random valid cases
{
```

```
  "input": {
    "commands": ["LRUCache", "put", "put", "get", "put", "get", "put", "get"],
    "args": [[2], [2, 1], [1, 1], [2], [4, 1], [1], [3, 1], [5]]
  },
  "output": [null, null, null, 1, null, 1, null, -1]
},

{
  "input": {
    "commands": ["LRUCache", "put", "put", "put", "put", "get", "get", "get"],
    "args": [[2], [1, 10], [2, 20], [3, 30], [4, 40], [2], [3], [4]]
  },
  "output": [null, null, null, null, null, -1, 30, 40]
},

{
  "input": {
    "commands": ["LRUCache", "put", "get", "put", "put", "get", "get"],
    "args": [[2], [1, 100], [1], [2, 200], [3, 300], [2], [1]]
  },
  "output": [null, null, 100, null, null, 200, -1]
},

// (You can continue with 20+ more randomized valid test cases based on above structure.)
{
  "input": {
    "commands": ["LRUCache", "put", "put", "put", "get", "put", "get", "get"],
    "args": [[2], [1, 1], [2, 2], [3, 3], [1], [4, 4], [2], [3]]
  },
  "output": [null, null, null, null, -1, null, -1, 3]
},
{
  "input": {
    "commands": ["LRUCache", "put", "get", "put", "get", "get"],
    "args": [[2], [2, 6], [2], [1, 5], [1], [2]]
  },
  "output": [null, null, 6, null, 5, 6]
},
{
  "input": {
    "commands": ["LRUCache", "put", "put", "put", "get", "put", "get"],
    "args": [[2], [2, 1], [1, 1], [3, 1], [1], [4, 1], [3]]
  },
  "output": [null, null, null, null, -1, null, 1]
},
{
  "input": {
    "commands": ["LRUCache", "put", "put", "get", "put", "put", "get"],
    "args": [[2], [1, 1], [2, 2], [2], [3, 3], [4, 4], [1]]
```

```
  },
  "output": [null, null, null, 2, null, null, -1]
},
{
  "input": {
    "commands": ["LRUCache", "put", "get", "put", "get", "put", "get", "get"],
    "args": [[2], [1, 1], [1], [2, 2], [2], [3, 3], [1], [3]]
  },
  "output": [null, null, 1, null, 2, null, -1, 3]
},
{
  "input": {
    "commands": ["LRUCache", "put", "put", "get", "put", "get", "put", "get"],
    "args": [[2], [2, 6], [3, 7], [2], [4, 5], [3], [5, 9], [4]]
  },
  "output": [null, null, null, 6, null, -1, null, 5]
},
{
  "input": {
    "commands": ["LRUCache", "put", "get", "put", "get", "put", "put", "get"],
    "args": [[3], [1, 1], [1], [2, 2], [2], [3, 3], [4, 4], [3]]
  },
  "output": [null, null, 1, null, 2, null, null, 3]
},
{
  "input": {
    "commands": ["LRUCache", "put", "put", "put", "get", "put", "get"],
    "args": [[2], [1, 2], [2, 3], [3, 4], [1], [4, 5], [2]]
  },
  "output": [null, null, null, null, -1, null, -1]
},
{
  "input": {
    "commands": ["LRUCache", "put", "put", "get", "get", "put", "put", "get", "get"],
    "args": [[2], [1, 1], [2, 2], [1], [2], [3, 3], [4, 4], [3], [4]]
  },
  "output": [null, null, null, 1, 2, null, null, 3, 4]
},
{
  "input": {
    "commands": ["LRUCache", "put", "get", "put", "get", "put", "get", "put", "get"],
    "args": [[2], [2, 1], [2], [1, 2], [2], [3, 3], [2], [4, 4], [3]]
  },
  "output": [null, null, 1, null, 1, null, -1, null, 3]
},
{
  "input": {
    "commands": ["LRUCache", "put", "get", "put", "get", "put", "get"],
    "args": [[1], [1, 1], [1], [2, 2], [1], [3, 3], [2]]
  },
```

```
  "output": [null, null, 1, null, -1, null, -1]
},
{
  "input": {
    "commands": ["LRUCache", "put", "put", "get", "put", "put", "get", "get"],
    "args": [[2], [1, 1], [2, 2], [1], [3, 3], [4, 4], [1], [3]]
  },
  "output": [null, null, null, 1, null, null, -1, 3]
},
{
  "input": {
    "commands": ["LRUCache", "put", "put", "put", "get", "put", "get", "put", "get"],
    "args": [[2], [1, 10], [2, 20], [3, 30], [1], [4, 40], [2], [5, 50], [4]]
  },
  "output": [null, null, null, null, -1, null, -1, null, 40]
},
{
  "input": {
    "commands": ["LRUCache", "put", "put", "get", "get", "put", "put", "get", "get"],
    "args": [[3], [1, 100], [2, 200], [1], [2], [3, 300], [4, 400], [3], [4]]
  },
  "output": [null, null, null, 100, 200, null, null, 300, 400]
},
{
  "input": {
    "commands": ["LRUCache", "put", "put", "get", "get", "put", "put", "get", "put", "get"],
    "args": [[3], [1, 1], [2, 2], [1], [2], [3, 3], [4, 4], [3], [5, 5], [4]]
  },
  "output": [null, null, null, 1, 2, null, null, 3, null, 4]
}



] }, { "id": 200, "title": "Number of Islands", "testcases": [ { "input": { "grid":
[ ["1","1","1","1","0"], ["1","1","0","1","0"], ["1","1","0","0","0"],
["0","0","0","0","0"] ] }, "output": 1 },

// Size-boundary (~10)
{ "input": { "grid": [["0"]] }, "output": 0 },
{ "input": { "grid": [["1"]] }, "output": 1 },
{ "input": { "grid": Array(1).fill(Array(300).fill("0")) }, "output": 0 },
{ "input": { "grid": Array(300).fill(Array(1).fill("1")) }, "output": 300 },
{ "input": { "grid": Array(300).fill(Array(300).fill("0")) }, "output": 0 },
{ "input": { "grid": Array(300).fill(Array(300).fill("1")) }, "output": 1 },
{ "input": { "grid": Array(150).fill(Array(300).fill("1")) }, "output": 1 },
{ "input": { "grid": Array(300).fill(Array(300).fill("1")).map((row, i) => row.map((cell, j)
=> ((i + j) % 2 === 0 ? "1" : "0"))) }, "output": 45000 },
{ "input": { "grid": Array(300).fill(Array(300).fill("1")).map((row, i) => row.map((cell, j)
=> (i === j ? "1" : "0"))) }, "output": 300 },

// Value-boundary (~7)
```

```
{ "input": { "grid": [["1","0","1","0","1"]] }, "output": 3 },
{ "input": { "grid": [["0","1","0","1","0"]] }, "output": 2 },
{ "input": { "grid": [["1","1","0","0","1","1"]] }, "output": 2 },
{ "input": { "grid": [["1","1"],["1","1"]] }, "output": 1 },
{ "input": { "grid": [["1","0"],["0","1"]] }, "output": 2 },
{ "input": { "grid": [["0","0"],["0","0"]] }, "output": 0 },
{ "input": { "grid": [["1","1","1"],["0","1","0"],["1","1","1"]] }, "output": 1 },

// Functional edge/corner (~7)
{ "input": { "grid": [["1","0","0","0","1"]] }, "output": 2 },
{ "input": { "grid": [["1"],["0"],["1"],["0"],["1"]] }, "output": 3 },
{ "input": { "grid": [["1","1","1","0","1","1","1"]] }, "output": 1 },
{ "input": { "grid":
[["1","1","0","0","0"],["1","1","0","0","0"],["0","0","1","0","0"],["0","0","0","1","1"]] },
"output": 3 },
{ "input": { "grid": [["1","0","1","0"],["0","1","0","1"]] }, "output": 4 },
{ "input": { "grid": [["1","0","1","0","1","0"]] }, "output": 3 },
{ "input": { "grid": [["0","0","0"],["0","0","0"],["0","0","0"]] }, "output": 0 },

// Other valid/random (~24)
{ "input": { "grid":
[["1","1","0","0"],["0","1","0","0"],["0","0","1","1"],["0","0","1","1"]] }, "output": 2 },
{ "input": { "grid": [["1","0","1"],["0","1","0"],["1","0","1"]] }, "output": 5 },
{ "input": { "grid":
[["1","1","0","0","0"],["1","1","0","0","0"],["0","0","1","0","0"],["0","0","0","1","1"]] },
"output": 3 },
{ "input": { "grid": [["1","0","0"],["0","1","1"],["0","0","0"]] }, "output": 2 },
{ "input": { "grid": [["1","1","1"],["0","1","0"],["1","1","1"]] }, "output": 1 },
{ "input": { "grid": [["0","1","0"],["1","0","1"],["0","1","0"]] }, "output": 5 },
{ "input": { "grid": [["1","1","0"],["0","1","1"],["1","0","1"]] }, "output": 2 },
{ "input": { "grid": [["0","0","0"],["1","1","1"],["0","0","0"]] }, "output": 1 },
{ "input": { "grid": [["1","0","0"],["1","0","0"],["1","1","1"]] }, "output": 1 },
{ "input": { "grid": [["0","1","0","1","0"],["1","0","1","0","1"],["0","1","0","1","0"]] },
"output": 9 },
{ "input": { "grid": [["1","0","0","1"],["0","0","0","0"],["1","0","0","1"]] }, "output":
4 },
{ "input": { "grid": [["1","1","1","1"],["1","1","1","1"],["1","1","1","1"]] }, "output":
1 },
{ "input": { "grid": [["1","0","0","1"],["0","1","1","0"],["1","0","0","1"]] }, "output":
5 },
{ "input": { "grid":
[["1","1","0","0","0","0"],["1","0","1","1","0","0"],["0","0","0","0","1","1"]] }, "output":
3 },
{ "input": { "grid": [["0","1","1","0"],["1","0","0","1"],["1","1","1","1"]] }, "output":
2 },
{ "input": { "grid": [["1","0","1"],["1","1","1"],["1","0","1"]] }, "output": 1 },
{ "input": { "grid": [["1","1","0","0","1"],["0","0","1","1","0"],["1","1","0","0","1"]] },
"output": 5 },
{ "input": { "grid":
[["1","1","1","0","1","1","1"],["1","0","1","0","1","0","1"],["1","1","1","1","1","1","1"]] }
```

```
, "output": 1 },
{ "input": { "grid":
[["1","1","0","1"],["1","0","0","1"],["0","0","1","0"],["1","1","1","1"]] }, "output": 3 },
{ "input": { "grid":
[["1","1","1","0","0","1"],["0","0","0","1","1","1"],["1","1","0","1","1","0"]] }, "output":
3 },
{ "input": { "grid": [["1","0","0","0","1"],["0","0","1","0","0"],["1","0","0","0","1"]] },
"output": 5 },
{ "input": { "grid":
[["0","1","0","0","1","0"],["1","0","1","0","0","1"],["0","1","0","1","0","0"]] }, "output":
9 },
{ "input": { "grid":
[["1","1","0","0","0"],["0","1","0","0","1"],["1","0","1","1","1"],["0","0","0","0","0"]] },
"output": 3 },
{ "input": { "grid":
[["0","0","0","0"],["1","1","0","0"],["0","0","1","1"],["0","0","1","1"]] }, "output": 2 }


] }, { "id": 22, "title": "Generate Parentheses", "testcases": [ { "input": { "n": 1 },
"output": ["()"] }, { "input": { "n": 2 }, "output": ["(())", "()()"] }, { "input": { "n":
3 }, "output": ["((()))","(()())","(())()","()(())","()()()"] }, { "input": { "n": 4 },
"output":
["(((())))","((()()))","((())())","((()))()","(()(()))","(()()())","(()())()","(())(())","(()
)()()","()((()))","()(()())","()(())()","()()(())","()()()()"] }, { "input": { "n": 5 },
"output":
["((((()))))","(((()())))","(((())()))","(((()))())","(((())))()","((()(())))","((()()()))","
(((()))())","(((()()))()","(((())(())","(((()))()()","((()((())))","((()(()()))","((()())()
)(((())))","(()((()()))","(()(())())","(()(()))()","(()()(()))","(()()()())","(()()())()","(())
((()))","(())(()())","(())(())()","(())()(())","(())()()()","()(((())))","()((()()))","()(()
)())","()((()))()","()(()())()","()(())()()","()()((()))","()()(()())","()()()(())","()()()()
))","()()(()())","()()(())()","()()()(())","()()()()()"] } ] }, { "id": 33, "title": "Search
in Rotated Sorted Array", "testcases": [ // Sample test case { "input": { "nums":
[4,5,6,7,0,1,2], "target": 0 }, "output": 4 },

// Size-boundary cases (~10)
{ "input": { "nums": [1], "target": 1 }, "output": 0 },
{ "input": { "nums": [1], "target": 0 }, "output": -1 },
{ "input": { "nums": [1,3], "target": 3 }, "output": 1 },
{ "input": { "nums": [1,3], "target": 1 }, "output": 0 },
{ "input": { "nums": [1,3], "target": 2 }, "output": -1 },
{ "input": { "nums": [2,3,4,5,1], "target": 1 }, "output": 4 },
{ "input": { "nums": [2,3,4,5,1], "target": 3 }, "output": 1 },
{ "input": { "nums": [1,2,3,4,5,6,7,8,9,10], "target": 6 }, "output": 5 },
{ "input": { "nums": [7,8,9,10,1,2,3,4,5,6], "target": 6 }, "output": 9 },

// Value-boundary cases (~7)
{ "input": { "nums": [-10000,-9999,-9998,-9997], "target": -9999 }, "output": 1 },
{ "input": { "nums": [9997,9998,9999,10000,-10000,-9999], "target": -10000 }, "output": 4 },
{ "input": { "nums": [-10000,0,10000], "target": 0 }, "output": 1 },
{ "input": { "nums": [-10000,0,10000], "target": 10000 }, "output": 2 },
```

```
{ "input": { "nums": [10000,-10000,-9999], "target": -9999 }, "output": 2 },
{ "input": { "nums": [0], "target": 10000 }, "output": -1 },
{ "input": { "nums": [10000], "target": -10000 }, "output": -1 },

// Functional edge/corner cases (~7)
{ "input": { "nums": [5,6,7,8,9,1,2,3,4], "target": 1 }, "output": 5 },
{ "input": { "nums": [5,6,7,8,9,1,2,3,4], "target": 5 }, "output": 0 },
{ "input": { "nums": [5,6,7,8,9,1,2,3,4], "target": 4 }, "output": 8 },
{ "input": { "nums": [3,1], "target": 1 }, "output": 1 },
{ "input": { "nums": [3,1], "target": 3 }, "output": 0 },
{ "input": { "nums": [3,1], "target": 2 }, "output": -1 },
{ "input": { "nums": [30,40,50,10,20], "target": 10 }, "output": 3 },

// Other valid/random cases (~24)
{ "input": { "nums": [4,5,6,7,8,1,2,3], "target": 8 }, "output": 4 },
{ "input": { "nums": [4,5,6,7,8,1,2,3], "target": 2 }, "output": 6 },
{ "input": { "nums": [4,5,6,7,8,1,2,3], "target": 10 }, "output": -1 },
{ "input": { "nums": [2,3,4,5,6,7,8,9,10,1], "target": 1 }, "output": 9 },
{ "input": { "nums": [2,3,4,5,6,7,8,9,10,1], "target": 9 }, "output": 7 },
{ "input": { "nums": [10,20,30,40,50,60,70,80,90,100], "target": 75 }, "output": -1 },
{ "input": { "nums": [70,80,90,100,10,20,30,40,50,60], "target": 40 }, "output": 7 },
{ "input": { "nums": [3,4,5,6,7,8,9,1,2], "target": 2 }, "output": 8 },
{ "input": { "nums": [3,4,5,6,7,8,9,1,2], "target": 3 }, "output": 0 },
{ "input": { "nums": [3,4,5,6,7,8,9,1,2], "target": 9 }, "output": 6 },
{ "input": { "nums": [3,4,5,6,7,8,9,1,2], "target": 8 }, "output": 5 },
{ "input": { "nums": [3,4,5,6,7,8,9,1,2], "target": 6 }, "output": 3 },
{ "input": { "nums": [3,4,5,6,7,8,9,1,2], "target": 10 }, "output": -1 },
{ "input": { "nums": [2,1], "target": 2 }, "output": 0 },
{ "input": { "nums": [2,1], "target": 1 }, "output": 1 },
{ "input": { "nums": [2,1], "target": 3 }, "output": -1 },
{ "input": { "nums": [1,2,3,4,5,6,7], "target": 5 }, "output": 4 },
{ "input": { "nums": [4,5,6,7,0,1,2], "target": 6 }, "output": 2 },
{ "input": { "nums": [4,5,6,7,0,1,2], "target": 7 }, "output": 3 },
{ "input": { "nums": [4,5,6,7,0,1,2], "target": 8 }, "output": -1 },
{ "input": { "nums": [7,0,1,2,3,4,5,6], "target": 3 }, "output": 4 },
{ "input": { "nums": [7,0,1,2,3,4,5,6], "target": 7 }, "output": 0 },
{ "input": { "nums": [7,0,1,2,3,4,5,6], "target": 0 }, "output": 1 },

// Additional sample
{ "input": { "nums": [1], "target": 0 }, "output": -1 }


] }, { "id": 238, "title": "Product of Array Except Self", "testcases": [ // Sample Test Case
{ "input": [1,2,3,4], "output": [24,12,8,6] },

// Size-boundary cases (~10)
{ "input": [1, 2], "output": [2, 1] },
{ "input": [-1, 2], "output": [2, -1] },
{ "input": [0, 1], "output": [1, 0] },
{ "input": [1, 0], "output": [0, 1] },
```

```
{ "input": [5, 5, 5], "output": [25, 25, 25] },
{ "input": [0, 0, 1], "output": [0, 0, 0] },
{ "input": [-1, -2, -3], "output": [6, 3, 2] },
{ "input": [2]*10, "output": [1024/2]*10 },
{ "input": [1]*100000, "output": [1]*100000 },

// Value-boundary cases (~7)
{ "input": [-30, -30, -30], "output": [900, 900, 900] },
{ "input": [30, 30, 30], "output": [900, 900, 900] },
{ "input": [-30, 1, 30], "output": [30, -900, -30] },
{ "input": [0, -30, 30], "output": [-900, 0, 0] },
{ "input": [30, 0, 0], "output": [0, 0, 0] },
{ "input": [-30]*1000, "output": [int(pow(-30,999))//-30]*1000 },
{ "input": [30]*1000, "output": [int(pow(30,999))//30]*1000 },

// Functional edge/corner (~7)
{ "input": [0, 1, 2, 3], "output": [6, 0, 0, 0] },
{ "input": [1, 2, 0, 3], "output": [0, 0, 6, 0] },
{ "input": [1, 2, 3, 0], "output": [0, 0, 0, 6] },
{ "input": [0, 0, 0], "output": [0, 0, 0] },
{ "input": [1, 1, 1, 1, 0], "output": [0, 0, 0, 0, 1] },
{ "input": [1, -1, 1, -1], "output": [-1, 1, -1, 1] },
{ "input": [10**5]*10, "output": [10**45]*10 },

// Other valid/random cases (~24)
{ "input": [2, 3, 4], "output": [12, 8, 6] },
{ "input": [3, 2, 1], "output": [2, 3, 6] },
{ "input": [1, 1, 1, 1], "output": [1, 1, 1, 1] },
{ "input": [2, -2, 2, -2], "output": [-8, 8, -8, 8] },
{ "input": [4, 5, 6, 7], "output": [210, 168, 140, 120] },
{ "input": [-5, 2, 3, 4], "output": [24, -60, -40, -30] },
{ "input": [1, 0, 3], "output": [0, 3, 0] },
{ "input": [10, 1, 10, 1], "output": [10, 100, 10, 100] },
{ "input": [1, 2, 3, 0, 5], "output": [0, 0, 0, 30, 0] },
{ "input": [9, 8, 7, 6, 5], "output": [1680, 1890, 2160, 2520, 3024] },
{ "input": [2, 2, 3, 4, 5], "output": [240, 240, 160, 120, 96] },
{ "input": [1]*10, "output": [1]*10 },
{ "input": [5]*20, "output": [int(5**19)]*20 },
{ "input": [2, 0, 2], "output": [0, 4, 0] },
{ "input": [-1, -1, 2, 2], "output": [4, 4, -2, -2] },
{ "input": [1, 2, 3, 4, 5, 6], "output": [720, 360, 240, 180, 144, 120] },
{ "input": [3, 1, 4, 1, 5, 9], "output": [180, 540, 135, 540, 108, 60] },
{ "input": [0, 1, 2, 0, 4], "output": [0, 0, 0, 0, 0] },
{ "input": [2, 3, 5, 7, 11, 13], "output": [15015, 10010, 6006, 4290, 2730, 2310] },
{ "input": [8, 0, 3, 0], "output": [0, 0, 0, 0] },
{ "input": [2, 3, 0, 5, 6], "output": [0, 0, 180, 0, 0] },
{ "input": [-2, 3, -4, 5], "output": [-60, 40, -30, 24] },
{ "input": [2, 4, 6, 8, 10], "output": [1920, 960, 640, 480, 384] }
```

```
] }, { "id": 560, "title": "Subarray Sum Equals K", "testcases": [ // Sample test cases
{ "input": { "nums": [1, 1, 1], "k": 2 }, "output": 2 }, { "input": { "nums": [1, 2, 3], "k":
3 }, "output": 2 },

// Size-boundary cases (~10)
{ "input": { "nums": [1], "k": 1 }, "output": 1 },
{ "input": { "nums": [1000], "k": 1000 }, "output": 1 },
{ "input": { "nums": [-1000], "k": -1000 }, "output": 1 },
{ "input": { "nums": [1, -1], "k": 0 }, "output": 1 },
{ "input": { "nums": [0, 0], "k": 0 }, "output": 3 },
{ "input": { "nums": [1, 1], "k": 1 }, "output": 2 },
{ "input": { "nums": [1]*100, "k": 100 }, "output": 1 },
{ "input": { "nums": [1]*100, "k": 50 }, "output": 51 },
{ "input": { "nums": [0]*20000, "k": 0 }, "output": 200010000 },

// Value-boundary cases (~7)
{ "input": { "nums": [1000, -1000, 1000], "k": 1000 }, "output": 2 },
{ "input": { "nums": [-1000, 1000], "k": 0 }, "output": 1 },
{ "input": { "nums": [-1000, -1000, -1000], "k": -2000 }, "output": 2 },
{ "input": { "nums": [1000, 1000, 1000], "k": 2000 }, "output": 2 },
{ "input": { "nums": [-1]*10, "k": -10 }, "output": 1 },
{ "input": { "nums": [1000]*20, "k": 1000 }, "output": 20 },
{ "input": { "nums": [-1000]*20, "k": -1000 }, "output": 20 },

// Functional edge/corner (~7)
{ "input": { "nums": [1, 2, 1, 2, 1], "k": 3 }, "output": 4 },
{ "input": { "nums": [0, 0, 0, 0], "k": 0 }, "output": 10 },
{ "input": { "nums": [3, 4, 7, 2, -3, 1, 4, 2], "k": 7 }, "output": 4 },
{ "input": { "nums": [1, 1, 1, 1], "k": 2 }, "output": 3 },
{ "input": { "nums": [1, -1, 0], "k": 0 }, "output": 3 },
{ "input": { "nums": [5, 5, 5, 5], "k": 10 }, "output": 3 },
{ "input": { "nums": [0]*100 + [1], "k": 1 }, "output": 1 },

// Other valid/random cases (~24)
{ "input": { "nums": [1, 2, 1, 2, 1], "k": 6 }, "output": 0 },
{ "input": { "nums": [1, 2, 3, 4, 5], "k": 9 }, "output": 2 },
{ "input": { "nums": [1, 2, 1, 2, 1, 3], "k": 3 }, "output": 5 },
{ "input": { "nums": [1, -1, 1, -1], "k": 0 }, "output": 4 },
{ "input": { "nums": [3, 4, 7, 2, -3, 1, 4, 2], "k": 7 }, "output": 4 },
{ "input": { "nums": [1]*5000, "k": 1 }, "output": 5000 },
{ "input": { "nums": [1]*5000 + [2]*5000, "k": 3 }, "output": 9999 },
{ "input": { "nums": [1, -1, 2, -2, 3, -3, 4], "k": 0 }, "output": 6 },
{ "input": { "nums": [10, 2, -2, -20, 10], "k": -10 }, "output": 3 },
{ "input": { "nums": [1, 2, 3, -3, -2, -1], "k": 3 }, "output": 4 },
{ "input": { "nums": [1, 1, 1, 1, 1], "k": 5 }, "output": 1 },
{ "input": { "nums": [2, 3, -2, 1, 2], "k": 4 }, "output": 2 },
{ "input": { "nums": [2, 2, 2, 2, 2], "k": 4 }, "output": 4 },
{ "input": { "nums": [1, 2, 3, 4, 5, 6, 7], "k": 7 }, "output": 2 },
{ "input": { "nums": [1, -1, 0, 1, -1], "k": 0 }, "output": 7 },
{ "input": { "nums": [-1, -1, 1], "k": 0 }, "output": 1 },
```

{ "input": { "nums": [-1, -1, 1, 1], "k": 0 }, "output": 2 },
{ "input": { "nums": [-1, 2, 3, -2, 1], "k": 3 }, "output": 2 },
{ "input": { "nums": [1, -1, 5, -2, 3], "k": 3 }, "output": 3 },
{ "input": { "nums": [1, 2, 3, 0, 3, 2, 1], "k": 6 }, "output": 4 },
{ "input": { "nums": [1000, -1000, 1000, -1000], "k": 0 }, "output": 4 },
{ "input": { "nums": [-1]*5000 + [1]*5000, "k": 0 }, "output": 0 },
{ "input": { "nums": [1, 2, 3, 4, 5], "k": 15 }, "output": 1 },
{ "input": { "nums": [1, 2, 3, 4, 5], "k": 0 }, "output": 0 }

] }, { "id": 56, "title": "Merge Intervals", "testcases": [// Sample test cases { "input": [[1,3],[2,6],[8,10],[15,18]], "output": [[1,6],[8,10],[15,18]] }, { "input": [[1,4],[4,5]], "output": [[1,5]] },

// Size-boundary cases (~10)
{ "input": [[1,2]], "output": [[1,2]] },
{ "input": [[0,0]], "output": [[0,0]] },
{ "input": [[0,1],[1,2]], "output": [[0,2]] },
{ "input": [[5,10],[6,7]], "output": [[5,10]] },
{ "input": [[1,5],[2,3]], "output": [[1,5]] },
{ "input": [[0,2],[3,5],[6,8],[9,10]], "output": [[0,2],[3,5],[6,8],[9,10]] },
{ "input": [[1,10],[2,3],[4,5],[6,7]], "output": [[1,10]] },
{ "input": [[1,10000],[5000,10001]], "output": [[1,10001]] },
{ "input": [[1,2]]*10000, "output": [[1,2]] },

// Value-boundary cases (~7)
{ "input": [[0,10000],[10000,10000]], "output": [[0,10000]] },
{ "input": [[0,0],[1,1],[2,2]], "output": [[0,0],[1,1],[2,2]] },
{ "input": [[0,10000]], "output": [[0,10000]] },
{ "input": [[0,1],[1,10000]], "output": [[0,10000]] },
{ "input": [[10000,10000],[9999,10000]], "output": [[9999,10000]] },
{ "input": [[0,5000],[5001,10000]], "output": [[0,5000],[5001,10000]] },
{ "input": [[0,1],[2,3],[4,5],[6,7]], "output": [[0,1],[2,3],[4,5],[6,7]] },

// Functional edge/corner (~7)
{ "input": [[1,3],[3,5],[5,7]], "output": [[1,7]] },
{ "input": [[1,10],[2,6],[8,9]], "output": [[1,10]] },
{ "input": [[2,3],[4,5],[6,7],[8,9],[1,10]], "output": [[1,10]] },
{ "input": [[5,5],[5,5],[5,5]], "output": [[5,5]] },
{ "input": [[1,4],[5,6],[7,8],[3,5]], "output": [[1,6],[7,8]] },
{ "input": [[0,1],[0,2],[0,3],[0,4]], "output": [[0,4]] },
{ "input": [[1,4],[0,4]], "output": [[0,4]] },

// Other valid/random cases (~24)
{ "input": [[1,5],[2,4],[3,6]], "output": [[1,6]] },
{ "input": [[1,4],[0,2],[3,5]], "output": [[0,5]] },
{ "input": [[1,4],[5,6]], "output": [[1,4],[5,6]] },
{ "input": [[2,3],[4,5],[6,7],[8,9]], "output": [[2,3],[4,5],[6,7],[8,9]] },
{ "input": [[5,10],[1,3],[2,6],[15,18]], "output": [[1,10],[15,18]] },
{ "input": [[1,2],[2,3],[3,4],[4,5]], "output": [[1,5]] },

```
{ "input": [[10,20],[1,5],[6,9]], "output": [[1,9],[10,20]] },
{ "input": [[1,2],[2,4],[6,7],[5,6]], "output": [[1,4],[5,7]] },
{ "input": [[0,1],[0,2],[0,3],[4,6]], "output": [[0,3],[4,6]] },
{ "input": [[0,10],[5,15],[20,25]], "output": [[0,15],[20,25]] },
{ "input": [[3,5],[1,2],[6,8]], "output": [[1,2],[3,5],[6,8]] },
{ "input": [[1,10],[2,6],[3,5],[7,9]], "output": [[1,10]] },
{ "input": [[2,3],[4,5],[6,7],[8,9],[1,10]], "output": [[1,10]] },
{ "input": [[6,8],[1,9],[2,4],[4,7]], "output": [[1,9]] },
{ "input": [[1,4],[0,0]], "output": [[0,0],[1,4]] },
{ "input": [[1,3],[2,4],[5,7],[6,8]], "output": [[1,4],[5,8]] },
{ "input": [[0,1],[2,3],[1,2]], "output": [[0,3]] },
{ "input": [[0,5],[3,6],[4,7],[6,10]], "output": [[0,10]] },
{ "input": [[1,10000],[2,9999],[3,9998]], "output": [[1,10000]] },
{ "input": [[0,2],[2,4],[4,6],[6,8]], "output": [[0,8]] },
{ "input": [[5,6],[6,7],[7,8],[1,3],[2,4]], "output": [[1,4],[5,8]] },
{ "input": [[1,4],[6,10],[5,6],[2,5]], "output": [[1,10]] },
{ "input": [[0,1],[2,3],[4,5],[6,7],[8,9],[10,11]], "output":
[[0,1],[2,3],[4,5],[6,7],[8,9],[10,11]] },
{ "input": [[2,3],[4,5],[6,7],[8,9],[0,10]], "output": [[0,10]] }


] }, { "id": 300, "title": "Longest Increasing Subsequence", "testcases": [ // Sample test
cases { "input": [10,9,2,5,3,7,101,18], "output": 4 }, { "input": [0,1,0,3,2,3], "output":
4 }, { "input": [7,7,7,7,7,7,7], "output": 1 },

// Size-boundary cases (~10)
{ "input": [1], "output": 1 },
{ "input": [1, 2], "output": 2 },
{ "input": [2, 1], "output": 1 },
{ "input": [1, 2, 3], "output": 3 },
{ "input": [3, 2, 1], "output": 1 },
{ "input": [5]*2500, "output": 1 },
{ "input": [i for i in range(2500)], "output": 2500 },
{ "input": [2500 - i for i in range(2500)], "output": 1 },
{ "input": [1]*1250 + [2]*1250, "output": 2 },
{ "input": [1, 3, 2]*833 + [3], "output": 3 },

// Value-boundary cases (~7)
{ "input": [10**4]*10, "output": 1 },
{ "input": [-10**4]*10, "output": 1 },
{ "input": [i for i in range(-10, 11)], "output": 21 },
{ "input": [10**4, -10**4], "output": 1 },
{ "input": [i for i in range(-10000, 10001, 800)], "output": 26 },
{ "input": [0, -1, -2, -3], "output": 1 },
{ "input": [-10000, -9999, 0, 9999, 10000], "output": 5 },

// Functional edge/corner (~7)
{ "input": [1, 3, 2], "output": 2 },
{ "input": [5, 6, 7, 1, 2, 3], "output": 3 },
{ "input": [3, 1, 4, 2], "output": 2 },
```

```
{ "input": [1, 3, 5, 4, 7], "output": 4 },
{ "input": [4, 10, 4, 3, 8, 9], "output": 3 },
{ "input": [2, 2, 2, 2, 2], "output": 1 },
{ "input": [1, 2, 4, 3], "output": 3 },

// Other valid/random cases (~24)
{ "input": [3, 4, -1, 0, 6, 2, 3], "output": 4 },
{ "input": [10, 22, 9, 33, 21, 50, 41, 60], "output": 5 },
{ "input": [1, 5, 2, 6, 3, 7, 4, 8], "output": 5 },
{ "input": [2, 2, 2, 1, 2], "output": 2 },
{ "input": [10, 11, 2, 3, 4, 5], "output": 5 },
{ "input": [1, 3, 5, 4, 7], "output": 4 },
{ "input": [1, 2, 4, 3, 5, 4, 7, 2], "output": 5 },
{ "input": [2, 2, 2, 2], "output": 1 },
{ "input": [0, 2, 6, 4, 5, 1, 7], "output": 4 },
{ "input": [1, 3, 2, 3, 1, 4], "output": 4 },
{ "input": [1, 100, 2, 99, 3, 98, 4, 97], "output": 4 },
{ "input": [1, 11, 2, 10, 3, 9, 4, 8], "output": 4 },
{ "input": [1, 2, 1, 2, 1, 2], "output": 2 },
{ "input": [3, 1, 2, 4, 3, 5], "output": 4 },
{ "input": [5, 6, 1, 2, 3], "output": 3 },
{ "input": [1, 9, 2, 8, 3, 7, 4, 6], "output": 4 },
{ "input": [9, 8, 7, 3, 2, 1], "output": 1 },
{ "input": [1, 3, 2, 4, 5, 3, 6], "output": 5 },
{ "input": [1, 5, 3, 6, 7], "output": 4 },
{ "input": [3, 2, 1, 4, 5, 6], "output": 4 },
{ "input": [1, 4, 3, 6, 2, 8], "output": 4 },
{ "input": [0, 1, 0, 3, 2, 3], "output": 4 },
{ "input": [7, 3, 5, 3, 6, 2, 7], "output": 4 },
{ "input": [3, 10, 2, 1, 20], "output": 3 }


] }, { "id": 1, "title": "Two Sum", "testcases": [ // Sample test cases { "input": { "nums":
[2,7,11,15], "target": 9 }, "output": [0,1] }, { "input": { "nums": [3,2,4], "target": 6 },
"output": [1,2] }, { "input": { "nums": [3,3], "target": 6 }, "output": [0,1] },

// Size-boundary cases (~10)
{ "input": { "nums": [1, 2], "target": 3 }, "output": [0, 1] },
{ "input": { "nums": [-1, 1], "target": 0 }, "output": [0, 1] },
{ "input": { "nums": [0, 0], "target": 0 }, "output": [0, 1] },
{ "input": { "nums": [1, 3, 5], "target": 8 }, "output": [1, 2] },
{ "input": { "nums": [10, 20, 30, 40], "target": 70 }, "output": [2, 3] },
{ "input": { "nums": [1]*10000, "target": 2 }, "output": [0, 1] },
{ "input": { "nums": [1,2,3,...,9999], "target": 19998 }, "output": [9997, 9998] },
{ "input": { "nums": [10**9, -10**9], "target": 0 }, "output": [0, 1] },

// Value-boundary cases (~7)
{ "input": { "nums": [-10**9, 10**9], "target": 0 }, "output": [0, 1] },
{ "input": { "nums": [-10**9, -10**9], "target": -2_000_000_000 }, "output": [0, 1] },
{ "input": { "nums": [10**9, 10**9], "target": 2_000_000_000 }, "output": [0, 1] },
```

```
{ "input": { "nums": [0, 10**9], "target": 10**9 }, "output": [0, 1] },
{ "input": { "nums": [-10**9, 0], "target": -10**9 }, "output": [0, 1] },
{ "input": { "nums": [999_999_999, 1], "target": 1_000_000_000 }, "output": [0, 1] },
{ "input": { "nums": [2, 3, 1_000_000_000, -999_999_998], "target": 2 }, "output": [0, 3] },

// Functional edge/corner (~7)
{ "input": { "nums": [1, 5, 1], "target": 2 }, "output": [0, 2] },
{ "input": { "nums": [0, 4, 3, 0], "target": 0 }, "output": [0, 3] },
{ "input": { "nums": [3, 2, 3], "target": 6 }, "output": [0, 2] },
{ "input": { "nums": [1, 2, 3, 4, 4], "target": 8 }, "output": [3, 4] },
{ "input": { "nums": [2, 5, 5, 11], "target": 10 }, "output": [1, 2] },
{ "input": { "nums": [1, 3, 4, 2], "target": 6 }, "output": [2, 3] },
{ "input": { "nums": [3, 2, 4], "target": 6 }, "output": [1, 2] },

// Other valid/random cases (~24)
{ "input": { "nums": [5, 75, 25], "target": 100 }, "output": [1, 2] },
{ "input": { "nums": [2, 15, 11, 7], "target": 9 }, "output": [0, 3] },
{ "input": { "nums": [1, 2, 3, 4], "target": 7 }, "output": [2, 3] },
{ "input": { "nums": [10, 20, 10, 40, 50, 60, 70], "target": 50 }, "output": [0, 3] },
{ "input": { "nums": [3, 2, 95, 4, -3], "target": 92 }, "output": [2, 4] },
{ "input": { "nums": [1, 9, 2, 11], "target": 10 }, "output": [0, 1] },
{ "input": { "nums": [2, 4, 6, 8], "target": 10 }, "output": [1, 3] },
{ "input": { "nums": [2, 7, 11, 15], "target": 18 }, "output": [1, 2] },
{ "input": { "nums": [1, 3, 4, 2], "target": 6 }, "output": [2, 3] },
{ "input": { "nums": [1, 2, 3, 4, 5, 6], "target": 11 }, "output": [4, 5] },
{ "input": { "nums": [1, 2, 5, 9, 12], "target": 14 }, "output": [2, 4] },
{ "input": { "nums": [3, 1, 4, 2], "target": 6 }, "output": [2, 3] },
{ "input": { "nums": [5, 2, 4, 6], "target": 10 }, "output": [2, 3] },
{ "input": { "nums": [2, 8, 12, 15], "target": 20 }, "output": [1, 2] },
{ "input": { "nums": [2, 7, 11, 15], "target": 26 }, "output": [2, 3] },
{ "input": { "nums": [4, 4, 1, 2], "target": 8 }, "output": [0, 1] },
{ "input": { "nums": [1, 2, 3, 4, 5, 6], "target": 3 }, "output": [0, 1] },
{ "input": { "nums": [4, 6, 5, 1], "target": 10 }, "output": [0, 1] },
{ "input": { "nums": [0, 1, 2, 3, 4, 5, 6, 7], "target": 13 }, "output": [6, 7] },
{ "input": { "nums": [10, 5, 2, 3], "target": 15 }, "output": [0, 1] },
{ "input": { "nums": [4, 1, 90], "target": 91 }, "output": [1, 2] },
{ "input": { "nums": [1, 2, 3, 4, 5], "target": 9 }, "output": [3, 4] },
{ "input": { "nums": [3, 1, 5, 7, 5, 9], "target": 10 }, "output": [2, 4] },
{ "input": { "nums": [1, 6, 7, 8, 15], "target": 23 }, "output": [3, 4] }



] }, { "id": 53, "title": "Maximum Subarray", "testcases": [ // Sample test cases { "input":
[-2,1,-3,4,-1,2,1,-5,4], "output": 6 }, { "input": [1], "output": 1 }, { "input": [5,4,-
1,7,8], "output": 23 },

// Size-boundary cases (~10)
{ "input": [-1], "output": -1 },
{ "input": [-10000], "output": -10000 },
{ "input": [10000], "output": 10000 },
{ "input": [1, 2], "output": 3 },
```

```
{ "input": [-1, -2], "output": -1 },
{ "input": [1, -2, 3], "output": 3 },
{ "input": [0]*100000, "output": 0 },
{ "input": [1]*100000, "output": 100000 },
{ "input": [-1]*100000, "output": -1 },
{ "input": [-10000]*99999 + [1], "output": 1 },

// Value-boundary cases (~7)
{ "input": [10000, -10000, 10000], "output": 10000 },
{ "input": [-10000, -10000], "output": -10000 },
{ "input": [10000, 10000], "output": 20000 },
{ "input": [0, 0, 0], "output": 0 },
{ "input": [-10**4, 0, 10**4], "output": 10000 },
{ "input": [10**4]*5, "output": 50000 },
{ "input": [-10**4]*5, "output": -10000 },

// Functional edge/corner (~7)
{ "input": [1, -1, 1, -1], "output": 1 },
{ "input": [-2, -3, 4, -1, -2, 1, 5, -3], "output": 7 },
{ "input": [-1, -2, -3, -4], "output": -1 },
{ "input": [1, -3, 2, 1, -1], "output": 3 },
{ "input": [-2, 1], "output": 1 },
{ "input": [2, -1, 2, 3, 4, -5], "output": 10 },
{ "input": [8, -19, 5, -4, 20], "output": 21 },

// Other valid/random cases (~24)
{ "input": [3, -2, 5, -1], "output": 6 },
{ "input": [-3, -2, -1], "output": -1 },
{ "input": [2, 3, -2, 4], "output": 7 },
{ "input": [-2, -1, -2, -3], "output": -1 },
{ "input": [4, -1, 2, 1], "output": 6 },
{ "input": [1, 2, 3, 4, 5], "output": 15 },
{ "input": [-1, 2, 3, -5, 4], "output": 4 },
{ "input": [5, -9, 6, -2, 3], "output": 7 },
{ "input": [10, -3, -4, 7, 6, 5, -4, -1], "output": 21 },
{ "input": [-2, 1, -3, 4, -1, 2, 1, -5, 4], "output": 6 },
{ "input": [-1, -2, -3, -4, 0], "output": 0 },
{ "input": [100, -90, 80, -70, 60], "output": 100 },
{ "input": [0, 0, 0, -1], "output": 0 },
{ "input": [1, -2, 3, 10, -4, 7, 2, -5], "output": 18 },
{ "input": [0, 1, -2, 3, 4, -5, 8], "output": 10 },
{ "input": [-2, -3, 4, -1, -2, 1, 5], "output": 7 },
{ "input": [0, 1, -1, 0], "output": 1 },
{ "input": [1, -1, 1, -1, 1, -1, 1], "output": 1 },
{ "input": [-1, 1, -1, 1, -1, 1, -1, 1], "output": 1 },
{ "input": [-3, -1, -2, -1], "output": -1 },
{ "input": [5, -2, 3, -1, 2], "output": 7 },
{ "input": [-1, -2, 3, 4], "output": 7 },
{ "input": [3, -1, 4, -1, 5, -9, 2], "output": 10 },
```

```
{ "input": [0]*50000 + [1]*50000, "output": 50000 }


] }, { "id": 121, "title": "Best Time to Buy and Sell Stock", "testcases": [ // Sample test
cases { "input": [7,1,5,3,6,4], "output": 5 }, { "input": [7,6,4,3,1], "output": 0 },

// Size-boundary cases (~10)
{ "input": [1], "output": 0 },
{ "input": [2, 1], "output": 0 },
{ "input": [1, 2], "output": 1 },
{ "input": [5]*100000, "output": 0 },
{ "input": [1]*50000 + [10]*50000, "output": 9 },
{ "input": [i for i in range(1, 100001)], "output": 99999 },
{ "input": [i for i in range(100000, 0, -1)], "output": 0 },
{ "input": [0]*99999 + [10000], "output": 10000 },
{ "input": [10**4]*99999 + [0], "output": 0 },
{ "input": [0, 1], "output": 1 },

// Value-boundary cases (~7)
{ "input": [0, 10**4], "output": 10000 },
{ "input": [10**4, 0], "output": 0 },
{ "input": [5000, 10000], "output": 5000 },
{ "input": [10000, 5000], "output": 0 },
{ "input": [10000, 10000], "output": 0 },
{ "input": [0]*100000, "output": 0 },
{ "input": [10**4]*100000, "output": 0 },

// Functional edge/corner (~7)
{ "input": [3, 3, 5, 0, 0, 3, 1, 4], "output": 4 },
{ "input": [1, 2, 3, 4, 5], "output": 4 },
{ "input": [2, 4, 1], "output": 2 },
{ "input": [1, 4, 2, 5, 3, 6], "output": 5 },
{ "input": [7, 1, 5, 3, 6, 4, 8], "output": 7 },
{ "input": [2, 1, 2, 0, 1], "output": 1 },
{ "input": [3, 2, 6, 5, 0, 3], "output": 4 },

// Other valid/random cases (~24)
{ "input": [1, 7, 5, 3, 6, 4], "output": 6 },
{ "input": [2, 4, 6, 1, 7], "output": 6 },
{ "input": [5, 10, 4, 6, 2, 8], "output": 6 },
{ "input": [3, 8, 6, 1, 5], "output": 5 },
{ "input": [1, 2, 3, 4, 5, 6], "output": 5 },
{ "input": [6, 5, 4, 3, 2, 1], "output": 0 },
{ "input": [7, 2, 5, 6, 1, 3], "output": 4 },
{ "input": [1, 1000, 2, 999], "output": 999 },
{ "input": [3, 2, 6, 5, 0, 3], "output": 4 },
{ "input": [2, 1, 2, 1, 0, 1, 2], "output": 2 },
{ "input": [9, 8, 6, 7, 6, 8], "output": 2 },
{ "input": [2, 4, 1, 3, 5], "output": 4 },
{ "input": [1, 1, 1, 1], "output": 0 },
```

```
{ "input": [2, 5, 7, 1, 4, 3, 1, 3], "output": 5 },
{ "input": [4, 3, 2, 1, 0, 1], "output": 1 },
{ "input": [1, 6, 2, 8, 4, 10], "output": 9 },
{ "input": [1, 2, 90, 10, 110], "output": 109 },
{ "input": [10, 22, 5, 75, 65, 80], "output": 75 },
{ "input": [12, 11, 13, 9, 12, 8, 14, 13, 15], "output": 7 },
{ "input": [90, 80, 70, 60, 50], "output": 0 },
{ "input": [5, 5, 5, 5, 5], "output": 0 },
{ "input": [3, 2, 6, 5, 0, 3], "output": 4 },
{ "input": [1, 2, 90, 10, 110], "output": 109 },
{ "input": [100, 180, 260, 310, 40, 535, 695], "output": 655 }


] }, { "id": 20, "title": "Valid Parentheses", "testcases": [ // Sample test cases { "input":
"()", "output": true }, { "input": "()[]{}", "output": true }, { "input": "(]", "output":
false }, { "input": "([])", "output": true },

// Size-boundary cases (~10)
{ "input": "(", "output": false },
{ "input": ")", "output": false },
{ "input": "[", "output": false },
{ "input": "{", "output": false },
{ "input": "}", "output": false },
{ "input": "]", "output": false },
{ "input": "()", "output": true },
{ "input": "((((()))))", "output": true },
{ "input": "({[()]})", "output": true },
{ "input": "({[)", "output": false },
{ "input": "()" * 5000, "output": true },

// Value-boundary cases (~7)
{ "input": "({{{{{{{{{{{{{{{{{{{{}}}}}}}}}}}}}}}}}})", "output": true },
{ "input": "((((((((((((((((((((()))))))))))))))))))))", "output": true },
{ "input": "((((((((((((((((((((()))))))))))))))))))))", "output": false },
{ "input": "{}{}{}{}{}{}{}{}{}{}" * 1000, "output": true },
{ "input": "[" * 5000 + "]" * 4999, "output": false },
{ "input": "[" * 5000 + "]" * 5000, "output": true },
{ "input": "(" * 3334 + ")" * 3333, "output": false },

// Functional edge/corner (~7)
{ "input": "({[({})]})", "output": true },
{ "input": "({[({[]})]})", "output": false },
{ "input": "(()[]{})", "output": true },
{ "input": "(()", "output": false },
{ "input": "{[}]", "output": false },
{ "input": "{([])}", "output": true },
{ "input": "[({})](]", "output": false },

// Other valid/random cases (~24)
{ "input": "([)]", "output": false },
```

{ "input": "({[]})", "output": true },
{ "input": "({[]})()", "output": true },
{ "input": "(((((((((((())))))))))))", "output": true },
{ "input": "(((((((((((()))))))))))", "output": false },
{ "input": "{}[()]", "output": true },
{ "input": "}{", "output": false },
{ "input": "[[[]]]", "output": true },
{ "input": "([[[{{{((()))}}}]]])", "output": true },
{ "input": "[[[[{", "output": false },
{ "input": "(()(()))", "output": true },
{ "input": "()(()))", "output": false },
{ "input": "[](()){}){}", "output": true },
{ "input": "[](()){}){", "output": false },
{ "input": "(()[]{}{[()]})", "output": true },
{ "input": "((()[]{()}))", "output": true },
{ "input": "([)", "output": false },
{ "input": "[([]])", "output": false },
{ "input": "[([]{})]", "output": true },
{ "input": "([[[{{((()))}}]]])", "output": true },
{ "input": "(((((((((((((((((())))))))))))))))))", "output": true },
{ "input": "((((((((((((((((((()))))))))))))))))))", "output": true },
{ "input":
"[[[]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
]]]]]]]]]]]", "output": true },
{ "input":
"(())))))))))))))))))))))))))))))))
))))))))))))))))))))))))))))", "output": true }

] }, { "id": 206, "title": "Reverse Linked List", "testcases": [// Sample test cases
{ "input": [1, 2, 3, 4, 5], "output": [5, 4, 3, 2, 1] }, { "input": [1, 2], "output": [2,
1] }, { "input": [], "output": [] },

// Size-boundary cases (~10)
{ "input": [1], "output": [1] },
{ "input": [0], "output": [0] },
{ "input": [5000], "output": [5000] },
{ "input": [-5000], "output": [-5000] },
{ "input": [1, 2, 3], "output": [3, 2, 1] },
{ "input": [0, 0, 0], "output": [0, 0, 0] },
{ "input": [1]*5000, "output": [1]*5000 },
{ "input": [i for i in range(1, 5001)], "output": [i for i in range(5000, 0, -1)] },

// Value-boundary cases (~7)
{ "input": [5000, 4999, -5000], "output": [-5000, 4999, 5000] },
{ "input": [-5000, 0, 5000], "output": [5000, 0, -5000] },
{ "input": [5000, 5000, 5000], "output": [5000, 5000, 5000] },
{ "input": [-5000, -5000, -5000], "output": [-5000, -5000, -5000] },
{ "input": [0, 1, 5000], "output": [5000, 1, 0] },
{ "input": [5000, 1, 0], "output": [0, 1, 5000] },

```
{ "input": [-5000, 0, 5000], "output": [5000, 0, -5000] },

// Functional edge/corner (~7)
{ "input": [1, 2, 2, 3], "output": [3, 2, 2, 1] },
{ "input": [5, 4, 3, 2, 1], "output": [1, 2, 3, 4, 5] },
{ "input": [1, 2, 1], "output": [1, 2, 1] },
{ "input": [1, 1, 1, 1], "output": [1, 1, 1, 1] },
{ "input": [2, -2, 2, -2], "output": [-2, 2, -2, 2] },
{ "input": [-1, -2, -3], "output": [-3, -2, -1] },
{ "input": [3, 3, 3], "output": [3, 3, 3] },

// Other valid/random cases (~24)
{ "input": [10, 20, 30], "output": [30, 20, 10] },
{ "input": [1, 2, 3, 4, 5, 6], "output": [6, 5, 4, 3, 2, 1] },
{ "input": [100, 200], "output": [200, 100] },
{ "input": [9, 8, 7, 6], "output": [6, 7, 8, 9] },
{ "input": [1, 3, 5, 7], "output": [7, 5, 3, 1] },
{ "input": [0, 1, 2, 3, 4], "output": [4, 3, 2, 1, 0] },
{ "input": [11, 22, 33, 44], "output": [44, 33, 22, 11] },
{ "input": [42], "output": [42] },
{ "input": [1, -1, 2, -2, 3], "output": [3, -2, 2, -1, 1] },
{ "input": [10, 0, -10], "output": [-10, 0, 10] },
{ "input": [123, 456, 789], "output": [789, 456, 123] },
{ "input": [7, 14, 21, 28, 35], "output": [35, 28, 21, 14, 7] },
{ "input": [99, 1, 0, -1, -99], "output": [-99, -1, 0, 1, 99] },
{ "input": [3, 3, 2, 2, 1, 1], "output": [1, 1, 2, 2, 3, 3] },
{ "input": [-2, 0, 2, -4], "output": [-4, 2, 0, -2] },
{ "input": [0]*10, "output": [0]*10 },
{ "input": [i for i in range(100, 0, -1)], "output": [i for i in range(1, 101)] },
{ "input": [1, 2, 1, 2, 1, 2], "output": [2, 1, 2, 1, 2, 1] },
{ "input": [9999, 8888, 7777], "output": [7777, 8888, 9999] },
{ "input": [-1, 0, 1], "output": [1, 0, -1] },
{ "input": [7, 7, 7, 7], "output": [7, 7, 7, 7] },
{ "input": [0, 1, 0, 1], "output": [1, 0, 1, 0] },
{ "input": [-100, 0, 100], "output": [100, 0, -100] }


] }, { "id": 21, "title": "Merge Two Sorted Lists", "testcases": [ // Sample test cases
{ "input": [[1,2,4], [1,3,4]], "output": [1,1,2,3,4,4] }, { "input": [[], []], "output":
[] }, { "input": [[], [0]], "output": [0] },

// Size-boundary cases (~10)
{ "input": [[1], []], "output": [1] },
{ "input": [[], [1]], "output": [1] },
{ "input": [[-100], [-100]], "output": [-100, -100] },
{ "input": [[-100]*50, [-100]*50], "output": [-100]*100 },
{ "input": [[1]*25, [2]*25], "output": [1]*25 + [2]*25 },
{ "input": [[1, 3, 5], [2, 4, 6]], "output": [1,2,3,4,5,6] },
{ "input": [[-100, -50, 0, 50, 100], []], "output": [-100, -50, 0, 50, 100] },
{ "input": [[], [-100, -50, 0, 50, 100]], "output": [-100, -50, 0, 50, 100] },
```

```
{ "input": [[-50]*50, [50]*50], "output": [-50]*50 + [50]*50 },

// Value-boundary cases (~7)
{ "input": [[-100, -50], [-100, -99]], "output": [-100, -100, -99, -50] },
{ "input": [[100, 100], [100]], "output": [100, 100, 100] },
{ "input": [[-100, 0, 100], [0, 100]], "output": [-100, 0, 0, 100, 100] },
{ "input": [[-100]*5, [100]*5], "output": [-100]*5 + [100]*5 },
{ "input": [[-100, -100, -100], []], "output": [-100, -100, -100] },
{ "input": [[0, 0, 0], [0, 0]], "output": [0, 0, 0, 0, 0] },
{ "input": [[-50], [-50]], "output": [-50, -50] },

// Functional edge/corner (~7)
{ "input": [[1, 3, 5, 7], [2, 4, 6, 8]], "output": [1,2,3,4,5,6,7,8] },
{ "input": [[1, 2, 3], [1, 2, 3]], "output": [1,1,2,2,3,3] },
{ "input": [[1, 2], [3, 4, 5, 6]], "output": [1,2,3,4,5,6] },
{ "input": [[3, 4, 5], [1, 2]], "output": [1,2,3,4,5] },
{ "input": [[1], [2, 3, 4, 5]], "output": [1,2,3,4,5] },
{ "input": [[1, 3, 5, 7, 9], [0, 2, 4, 6, 8, 10]], "output": [0,1,2,3,4,5,6,7,8,9,10] },
{ "input": [[5, 6, 7], [1, 2, 3]], "output": [1,2,3,5,6,7] },

// Other valid/random cases (~24)
{ "input": [[1,3,5], [2,4,6,8,10]], "output": [1,2,3,4,5,6,8,10] },
{ "input": [[-5, -3, -1], [-6, -4, -2]], "output": [-6, -5, -4, -3, -2, -1] },
{ "input": [[0,1,2], [0,2,3,4]], "output": [0,0,1,2,2,3,4] },
{ "input": [[7,9], [1,2,3,4,5,6,8]], "output": [1,2,3,4,5,6,7,8,9] },
{ "input": [[-1, 0, 3], [-2, 2, 4]], "output": [-2, -1, 0, 2, 3, 4] },
{ "input": [[1,2,4,5], [3,6]], "output": [1,2,3,4,5,6] },
{ "input": [[1], [2]], "output": [1,2] },
{ "input": [[2], [1]], "output": [1,2] },
{ "input": [[5,6,7,8], [1,2,3,4]], "output": [1,2,3,4,5,6,7,8] },
{ "input": [[10, 20, 30], [15, 25]], "output": [10,15,20,25,30] },
{ "input": [[0,2,4,6], [1,3,5,7]], "output": [0,1,2,3,4,5,6,7] },
{ "input": [[1]*50, [2]*50], "output": [1]*50 + [2]*50 },
{ "input": [[-100, 0, 100], [-50, 50]], "output": [-100, -50, 0, 50, 100] },
{ "input": [[3], [2, 2, 2]], "output": [2, 2, 2, 3] },
{ "input": [[1, 3, 5, 7, 9], [2, 4, 6, 8, 10]], "output": [1,2,3,4,5,6,7,8,9,10] },
{ "input": [[-1, 1, 3], [-2, 2, 4]], "output": [-2, -1, 1, 2, 3, 4] },
{ "input": [[100], [-100]], "output": [-100, 100] },
{ "input": [[-10, -5, 0], [5, 10, 15]], "output": [-10, -5, 0, 5, 10, 15] },
{ "input": [[-3, -1, 2, 4], [-2, 0, 3]], "output": [-3, -2, -1, 0, 2, 3, 4] },
{ "input": [[10,20,30,40], [5,15,25,35]], "output": [5,10,15,20,25,30,35,40] },
{ "input": [[1,2,3], []], "output": [1,2,3] },
{ "input": [[], [1,2,3]], "output": [1,2,3] },
{ "input": [[-10, -5, 0], []], "output": [-10, -5, 0] }


] }, { "id": 70, "title": "Climbing Stairs", "testcases": [ // Sample test cases { "input":
2, "output": 2 }, { "input": 3, "output": 3 },
```

```
// Size-boundary cases (~10)
{ "input": 1, "output": 1 },
{ "input": 2, "output": 2 },
{ "input": 3, "output": 3 },
{ "input": 4, "output": 5 },
{ "input": 5, "output": 8 },
{ "input": 10, "output": 89 },
{ "input": 20, "output": 10946 },
{ "input": 30, "output": 1346269 },
{ "input": 45, "output": 1836311903 },

// Value-boundary cases (~7)
{ "input": 6, "output": 13 },
{ "input": 7, "output": 21 },
{ "input": 8, "output": 34 },
{ "input": 9, "output": 55 },
{ "input": 11, "output": 144 },
{ "input": 12, "output": 233 },
{ "input": 13, "output": 377 },

// Functional edge/corner (~7)
{ "input": 14, "output": 610 },
{ "input": 15, "output": 987 },
{ "input": 16, "output": 1597 },
{ "input": 17, "output": 2584 },
{ "input": 18, "output": 4181 },
{ "input": 19, "output": 6765 },
{ "input": 21, "output": 17711 },

// Other valid/random cases (~24)
{ "input": 22, "output": 28657 },
{ "input": 23, "output": 46368 },
{ "input": 24, "output": 75025 },
{ "input": 25, "output": 121393 },
{ "input": 26, "output": 196418 },
{ "input": 27, "output": 317811 },
{ "input": 28, "output": 514229 },
{ "input": 29, "output": 832040 },
{ "input": 31, "output": 2178309 },
{ "input": 32, "output": 3524578 },
{ "input": 33, "output": 5702887 },
{ "input": 34, "output": 9227465 },
{ "input": 35, "output": 14930352 },
{ "input": 36, "output": 24157817 },
{ "input": 37, "output": 39088169 },
{ "input": 38, "output": 63245986 },
{ "input": 39, "output": 102334155 },
{ "input": 40, "output": 165580141 },
{ "input": 41, "output": 267914296 },
```

```
{ "input": 42, "output": 433494437 },
{ "input": 43, "output": 701408733 },
{ "input": 44, "output": 1134903170 },
{ "input": 5, "output": 8 },
{ "input": 9, "output": 55 }


] }, { "id": 136, "title": "Single Number", "testcases": [ // Sample test cases { "input":
[2, 2, 1], "output": 1 }, { "input": [4, 1, 2, 1, 2], "output": 4 }, { "input": [1],
"output": 1 },

// Size-boundary cases (~10)
{ "input": [1, 1, 2], "output": 2 },
{ "input": [3, 2, 2], "output": 3 },
{ "input": [0, 0, 1], "output": 1 },
{ "input": [30000, 30000, -30000], "output": -30000 },
{ "input": [-30000, -30000, 1], "output": 1 },
{ "input": [0], "output": 0 },
{ "input": [5]*2 + [99], "output": 99 },
{ "input": [9999]*2 + [12345], "output": 12345 },

// Value-boundary cases (~7)
{ "input": [30000]*2 + [-1], "output": -1 },
{ "input": [-30000]*2 + [0], "output": 0 },
{ "input": [0]*2 + [30000], "output": 30000 },
{ "input": [-1]*2 + [-30000], "output": -30000 },
{ "input": [123, 123, -30000], "output": -30000 },
{ "input": [-30000, 1, -30000], "output": 1 },
{ "input": [30000, -1, 30000], "output": -1 },

// Functional edge/corner (~7)
{ "input": [7, 7, 8, 8, 9], "output": 9 },
{ "input": [1, 1, 2, 2, 3], "output": 3 },
{ "input": [5, 5, 6, 6, 7, 7, 8], "output": 8 },
{ "input": [10, 10, 20, 20, 30, 40, 40], "output": 30 },
{ "input": [0, 1, 1, 0, 2], "output": 2 },
{ "input": [-2, -2, -3], "output": -3 },
{ "input": [2, 2, -3, -3, 0], "output": 0 },

// Other valid/random cases (~24)
{ "input": [1, 2, 1, 2, 3], "output": 3 },
{ "input": [9, 8, 7, 9, 8], "output": 7 },
{ "input": [11, 22, 11, 33, 22], "output": 33 },
{ "input": [101, 202, 303, 202, 101], "output": 303 },
{ "input": [-1, -1, 9999], "output": 9999 },
{ "input": [7, 7, 8, 8, 6, 6, 5], "output": 5 },
{ "input": [1]*1000 + [99], "output": 99 },
{ "input": [i for i in range(1, 5000)]*2 + [999999], "output": 999999 },
{ "input": [3, 3, 4, 4, 5, 5, 6], "output": 6 },
{ "input": [12345, 54321, 12345], "output": 54321 },
```

{ "input": [9999, 8888, 8888], "output": 9999 },
{ "input": [10, 10, 20], "output": 20 },
{ "input": [7, 3, 3], "output": 7 },
{ "input": [9, 9, 1], "output": 1 },
{ "input": [11, 22, 33, 22, 11], "output": 33 },
{ "input": [6, 5, 6], "output": 5 },
{ "input": [999, 999, 1000], "output": 1000 },
{ "input": [0, 1, 0], "output": 1 },
{ "input": [5, 5, 7, 7, 9], "output": 9 },
{ "input": [42]*2 + [17], "output": 17 },
{ "input": [2, 3, 2], "output": 3 },
{ "input": [8, 8, 6, 6, 4], "output": 4 },
{ "input": [10101]*2 + [20202], "output": 20202 },
{ "input": [99999]*2 + [1], "output": 1 }

] }, { "id": 101, "title": "Symmetric Tree", "testcases": [// Sample test cases { "input": [1,2,2,3,4,4,3], "output": true }, { "input": [1,2,2,null,3,null,3], "output": false },

// Size-boundary cases (~10)
{ "input": [1], "output": true },
{ "input": [1,2,2], "output": true },
{ "input": [1,2,2,null,3,3,null], "output": true },
{ "input": [1,2,2,null,3,null,3], "output": false },
{ "input": [1,2,2,3,null,null,3], "output": true },
{ "input": [1,2,2,3,4,4,5], "output": false },
{ "input": [1,2,2,3,4,4,3,5,null,null,5,null,null,null], "output": true },
{ "input": [1,2,2,3,4,4,3,5,6,6,5,5,6,6,5], "output": true },
{ "input": [1,2,2,3,4,4,3,null,null,null,null,null,null,null,8], "output": false },

// Value-boundary cases (~7)
{ "input": [0], "output": true },
{ "input": [0, -100, -100], "output": true },
{ "input": [0, -100, -100, -100], "output": false },
{ "input": [0, 100, 100], "output": true },
{ "input": [1,2,2,100,-100,-100,100], "output": true },
{ "input": [1,2,2,100,null,null,101], "output": false },
{ "input": [0,0,0,null,1,null,1], "output": false },

// Functional edge/corner (~7)
{ "input": [1,2,2,3,4,4,3], "output": true },
{ "input": [1,2,2,null,3,null,3], "output": false },
{ "input": [1,2,2,null,3,3,null], "output": true },
{ "input": [1,null,2], "output": false },
{ "input": [1,2,null], "output": false },
{ "input": [1,2,2,null,null,2,null], "output": false },
{ "input": [1,2,2,3,4,4,3,null,null,5,6,6,5], "output": true },

// Other valid/random cases (~24)
{ "input": [1,2,2,3,4,4,3,5,6,6,5], "output": true },

{ "input": [1,2,2,3,4,4,3,null,null,null,null,null,null,7,7], "output": true },
{ "input": [1,2,2,null,null,2,null], "output": false },
{ "input": [1,2,2,null,3,null,4], "output": false },
{ "input": [1,2,2,null,null,3,3], "output": true },
{ "input": [1,2,2,3,null,null,3,null,4,null,4], "output": true },
{ "input": [1,2,2,null,3,3,null,null,5,5], "output": true },
{ "input": [1,2,2,3,null,null,3,4,null,null,4], "output": true },
{ "input": [1,2,2,3,null,null,3,null,null,4,4], "output": true },
{ "input": [1,2,2,null,4,4,null,3,null,null,3], "output": true },
{ "input": [1,2,2,3,null,null,3,null,4,null,4], "output": true },
{ "input": [1,2,2,null,3,null,3,null,4,null,4], "output": true },
{ "input": [1,2,2,null,3,null,3,null,null,4,4], "output": true },
{ "input": [1,2,2,null,3,null,4], "output": false },
{ "input": [1,2,2,null,3,4,null], "output": false },
{ "input": [1,2,2,null,null,null,3], "output": false },
{ "input": [1,2,2,3,null,null,3,4,null,null,4], "output": true },
{ "input": [1,2,2,3,4,4,3,5,6,6,5], "output": true },
{ "input": [1,2,2,3,null,null,3,4,null,null,4,5,null,null,5], "output": true },
{ "input": [1,2,2,3,4,4,3,5,null,null,5], "output": true },
{ "input": [1,2,2,3,null,null,3,null,null,4,4], "output": true },
{ "input": [1,2,2,3,4,4,3,5,6,6,5,null,null,null,null,7,8,8,7], "output": true },
{ "input": [1,2,2,3,null,null,3,4,null,null,4,5,null,null,5,6,null,null,6], "output": true }

] }, { "id": 160, "title": "Intersection of Two Linked Lists", "testcases": [// Sample test
cases { "intersectVal": 8, "listA": [4,1,8,4,5], "listB": [5,6,1,8,4,5], "skipA": 2, "skipB":
3, "output": "Intersected at '8'" }, { "intersectVal": 2, "listA": [1,9,1,2,4], "listB":
[3,2,4], "skipA": 3, "skipB": 1, "output": "Intersected at '2'" }, { "intersectVal": 0,
"listA": [2,6,4], "listB": [1,5], "skipA": 3, "skipB": 2, "output": "No intersection" },

// Size-boundary cases (~10)
{
  "intersectVal": 1,
  "listA": [1],
  "listB": [1],
  "skipA": 0,
  "skipB": 0,
  "output": "Intersected at '1'"
},
{
  "intersectVal": 0,
  "listA": [1],
  "listB": [2],
  "skipA": 1,
  "skipB": 1,
  "output": "No intersection"
},
{
  "intersectVal": 50000,
  "listA": [50000],

```
  "listB": [99999,50000],
  "skipA": 0,
  "skipB": 1,
  "output": "Intersected at '50000'"
},
{
  "intersectVal": 30000,
  "listA": [1,2,3,30000,100,200],
  "listB": [7,8,9,30000,100,200],
  "skipA": 3,
  "skipB": 3,
  "output": "Intersected at '30000'"
},
{
  "intersectVal": 0,
  "listA": [1]*1000,
  "listB": [2]*1000,
  "skipA": 1000,
  "skipB": 1000,
  "output": "No intersection"
},
{
  "intersectVal": 42,
  "listA": [1,2,42,43],
  "listB": [3,42,43],
  "skipA": 2,
  "skipB": 1,
  "output": "Intersected at '42'"
},
{
  "intersectVal": 0,
  "listA": [],
  "listB": [],
  "skipA": 0,
  "skipB": 0,
  "output": "No intersection"
},

// Value-boundary cases (~7)
{
  "intersectVal": 1,
  "listA": [1],
  "listB": [2,1],
  "skipA": 0,
  "skipB": 1,
  "output": "Intersected at '1'"
},
{
  "intersectVal": 100000,
  "listA": [3,100000],
```

```
  "listB": [4,5,6,100000],
  "skipA": 1,
  "skipB": 3,
  "output": "Intersected at '100000'"
},
{
  "intersectVal": 99999,
  "listA": [1,2,3,99999],
  "listB": [99999],
  "skipA": 3,
  "skipB": 0,
  "output": "Intersected at '99999'"
},
{
  "intersectVal": 0,
  "listA": [1,2,3],
  "listB": [4,5,6],
  "skipA": 3,
  "skipB": 3,
  "output": "No intersection"
},
{
  "intersectVal": 105,
  "listA": [101,102,105],
  "listB": [103,104,105],
  "skipA": 2,
  "skipB": 2,
  "output": "Intersected at '105'"
},
{
  "intersectVal": 30000,
  "listA": [30000],
  "listB": [1,2,30000],
  "skipA": 0,
  "skipB": 2,
  "output": "Intersected at '30000'"
},
{
  "intersectVal": 0,
  "listA": [1]*3000,
  "listB": [2]*3000,
  "skipA": 3000,
  "skipB": 3000,
  "output": "No intersection"
},

// Functional edge/corner (~7)
{
  "intersectVal": 9,
  "listA": [1,2,3,9,10],
```

```
  "listB": [9,10],
  "skipA": 3,
  "skipB": 0,
  "output": "Intersected at '9'"
},
{
  "intersectVal": 7,
  "listA": [1,7],
  "listB": [2,3,7],
  "skipA": 1,
  "skipB": 2,
  "output": "Intersected at '7'"
},
{
  "intersectVal": 50,
  "listA": [50],
  "listB": [50],
  "skipA": 0,
  "skipB": 0,
  "output": "Intersected at '50'"
},
{
  "intersectVal": 0,
  "listA": [],
  "listB": [1],
  "skipA": 0,
  "skipB": 0,
  "output": "No intersection"
},
{
  "intersectVal": 13,
  "listA": [5,13,14,15],
  "listB": [6,7,8,13,14,15],
  "skipA": 1,
  "skipB": 3,
  "output": "Intersected at '13'"
},
{
  "intersectVal": 88,
  "listA": [88],
  "listB": [77,88],
  "skipA": 0,
  "skipB": 1,
  "output": "Intersected at '88'"
},
{
  "intersectVal": 0,
  "listA": [1],
  "listB": [2],
  "skipA": 1,
```

```
  "skipB": 1,
  "output": "No intersection"
},

// Other valid/random cases (~24)
{
  "intersectVal": 3,
  "listA": [1,2,3,4],
  "listB": [5,6,3,4],
  "skipA": 2,
  "skipB": 2,
  "output": "Intersected at '3'"
},
{
  "intersectVal": 15,
  "listA": [7,8,15,16],
  "listB": [9,10,15,16],
  "skipA": 2,
  "skipB": 2,
  "output": "Intersected at '15'"
},
{
  "intersectVal": 21,
  "listA": [5,6,7,21],
  "listB": [21],
  "skipA": 3,
  "skipB": 0,
  "output": "Intersected at '21'"
},
{
  "intersectVal": 33,
  "listA": [1,2,33,34],
  "listB": [10,20,30,33,34],
  "skipA": 2,
  "skipB": 3,
  "output": "Intersected at '33'"
},
{
  "intersectVal": 0,
  "listA": [1,2,3],
  "listB": [4,5,6],
  "skipA": 3,
  "skipB": 3,
  "output": "No intersection"
},
{
  "intersectVal": 40,
  "listA": [40],
  "listB": [40],
  "skipA": 0,
```

```
  "skipB": 0,
  "output": "Intersected at '40'"
},
{
  "intersectVal": 14,
  "listA": [9,10,11,12,13,14],
  "listB": [14],
  "skipA": 5,
  "skipB": 0,
  "output": "Intersected at '14'"
},
{
  "intersectVal": 0,
  "listA": [1],
  "listB": [2],
  "skipA": 0,
  "skipB": 0,
  "output": "No intersection"
},
{
  "intersectVal": 77,
  "listA": [10,20,77,78],
  "listB": [77,78],
  "skipA": 2,
  "skipB": 0,
  "output": "Intersected at '77'"
},
{
  "intersectVal": 999,
  "listA": [999],
  "listB": [1,2,999],
  "skipA": 0,
  "skipB": 2,
  "output": "Intersected at '999'"
}


] },

}
```