68.8

Patricia Roque

Other

PDF generated at: 24 Sep 2024 20:52:08 UTC

View this report on HackerRank

Score

68.8% • 55 / 80

scored in TIP102: Unit 1 Version B (Standard) - Fall 2024 in 76 min 12 sec on 24 Sep 2024 12:33:55 PDT

Candidate Information

Email	mhyles@bu.edu
Test	TIP102: Unit 1 Version B (Standard) - Fall 2024
Candidate Packet	View
Taken on	24 Sep 2024 12:33:55 PDT
Time taken	76 min 12 sec/ 90 min
Univ/College Name	Boston University
Personal Member ID	111572
Email Address with CodePath	mhyles@bu.edu
Github username with CodePath	itsmhyles
Invited by	CodePath

Skill Distribution

There is no associated skills data that can be shown for this assessment

Tags Distribution

There is no associated tags data that can be shown for this assessment

Questions

Coding Questions • 40.0 / 60.0

Status	No.	Question	Time Taken	Skill	Score
✓	1	Check For X Coding	6 min 4 sec	-	20/20
✓	2	First Repeating Substring Coding	16 min 24 sec	-	20/20

⊗	3	Special Array Coding	47 min 7 sec	-	0/20

Multiple Choice + Debugging • 15.0 / 20.0

Status	No.	Question	Time Taken	Skill	Score
✓	4	What is the output of the following code snippet? Multiple Choice	23 sec	-	5/5
✓	5	What is the output of the following code snippet? Multiple Choice	46 sec	-	5/5
⊗	6	What is the output? Multiple Choice	25 sec	-	0/5
✓	7	Find the bug! Coding	1 min 40 sec	-	5/5

1. Check For X

✓ Correct

Coding

Question description

Given a list of integers `nums`, return `True` if there are at least `x` numbers in `nums` that are greater than or equal to the length of the `nums`. The value of `x` is the length of the list.

Return `False` otherwise.

Example 1:
Input: lst = [1, 2, 3, 4]
Expected Output: False

Example 2:
Input: [4, 4, 4, 4]
Expected Output: True

Example 3:
Input: [5, 6, 7, 8]
Expected Output: True

Candidate's Solution

Language used: **Python 3**

```python
#!/bin/python3

import math
import os
import random
import re
import sys
import ast


#
# Complete the 'check_list' function below.
#
# The function is expected to return a BOOLEAN.
# The function accepts INTEGER_ARRAY nums as parameter.
#

def check_list(nums):
    # Write your code here
    '''
    nums = list of integer
    return boolean
    if each x in nums >= len(list)
    '''
    x = sum(1 for num in nums if num >= len(nums))
    return x == len(nums)

```

```
if __name__ == '__main__':
    input_lines = sys.stdin.read().strip().splitlines()

    for input_str in input_lines:
        if input_str.strip() == "":
            continue

        try:
            nums = ast.literal_eval(input_str)

            result = check_list(nums)

            print(result)
        except (ValueError, SyntaxError):
            print("Error: Invalid input")
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Basic Case	Easy	Hidden	Success	0	0.0753 sec	11.1 KB
All Elements Equal to Length	Easy	Hidden	Success	0	0.0452 sec	11.1 KB
Numbers Greater Than or Equal to Length	Easy	Hidden	Success	0	0.0405 sec	10.8 KB
No Elements Greater Than or Equal to Length	Easy	Hidden	Success	0	0.0433 sec	11.2 KB
Testcase 5	Easy	Hidden	Success	0	0.045 sec	11.1 KB
Empty List	Easy	Hidden	Success	0	0.0415	10.9 KB

					sec	
Mixed Values with Insufficient Count	Easy	Hidden	Success	0	0.0386 sec	11 KB
Testcase 7	Easy	Hidden	Success	0	0.0385 sec	10.9 KB
Pass/Fail Case	Easy	Hidden	Success	20	0.0417 sec	11.2 KB

No comments.

2. First Repeating Substring

Correct

Coding

Question description

Given a string s, find the first substring of length k that appears exactly twice in the string and return its starting index. If no such substring exists, return -1

```
Example usage:
Input: s = "abcabcabc" k = 3
Expected Output: 0

Example Usage 2:
Input: s = "banana" k = 2
Expected Output: 1
```

Candidate's Solution

Language used: **Python 3**

```python
#!/bin/python3

import math
import os
import random
import re
import sys
import ast


#
# Complete the 'first_repeating_substring' function below.
#
# The function is expected to return an INTEGER.
# The function accepts following parameters:
#  1. STRING s
#  2. INTEGER k
#

def first_repeating_substring(s, k):
    # Write your code here
    # given: string s, k = integer ?
    '''
    objective:
    find substring of length k that appears exactly twice
    return index, else return -1
    '''
    if k >= 1:
        for i in range(len(s)):
            if s[i:(k+i)] == s[(k+i):((k*2)+i)]:
               return i
        return -1



if __name__ == '__main__':
    input_str = sys.stdin.read().strip().splitlines()

    for line in input_str:
        s, k = line.split(', ')
        s = s.strip('"')
        k = int(k)
        result = first_repeating_substring(s, k)
        print(result)
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Basic Case with No Repetition	Easy	Hidden	Success	0	0.0411 sec	11.1 KB
Test Case with Overlapping Substrings	Easy	Hidden	Success	0	0.0365 sec	11 KB
Test Case with No Repeating Substring	Easy	Hidden	Success	0	0.0401 sec	10.9 KB
Test Case where k is Greater than String Length	Easy	Hidden	Success	0	0.0471 sec	11 KB
Test Case with a Single Occurrence of Substring	Easy	Hidden	Success	0	0.0461 sec	11.2 KB
Test case with empty string	Easy	Hidden	Success	0	0.0427 sec	11 KB
Test case with a single character string	Easy	Hidden	Success	0	0.0432 sec	11.2 KB
Pass/Fail Case	Easy	Hidden	Success	20	0.0374 sec	11 KB

No comments.

3. Special Array

Incorrect

Coding

Question description

You are given a list `nums` of non-negative integers. `nums` is considered **special** if there exists a number `x` such that there are **exactly** `x` numbers in `nums` that are **greater than or equal to** `x`.

Notice that `x` **does not** have to be an element in `nums`.

Return `x` *if the list is **special**, otherwise, return* `-1`. It can be proven that if `nums` is special, the value for `x` is **unique**.
Note: Students will need to use the built-in function sort().

Example 1:
Input: nums = [3,5]
Output: 2
Explanation: There are 2 values (3 and 5) that are greater than or equal to 2.

Example 2:
Input: nums = [0,0]
Output: -1
Explanation: No numbers fit the criteria for x.
If x = 0, there should be 0 numbers >= x, but there are 2.
If x = 1, there should be 1 number >= x, but there are 0.
If x = 2, there should be 2 numbers >= x, but there are 0.
x cannot be greater since there are only 2 numbers in nums.

Example 3:
Input: nums = [0,4,3,0,4]
Output: 3
Explanation: There are 3 values that are greater than or equal to 3.

Candidate's Solution

Language used: **Python 3**

```
#!/bin/python3
```

```

import math
import os
import random
import re
import sys
import ast



#
# Complete the 'special_array' function below.
#
# The function is expected to return an INTEGER.
# The function accepts INTEGER_ARRAY nums as parameter.
#

def special_array(nums):
    # Write your code here
    right = len(nums)-1
    special_number = 0

    if not nums:
        return -1


    for num in range(len(nums)):
        special_number = 0
        right = len(nums)-1
        while right >= 0:
            if nums[right]>= num+1:
                special_number +=1
            right -= 1
        if special_number == num+1:
            return num+1
    return -1



if __name__ == '__main__':
    input_lines = sys.stdin.read().strip().splitlines()

    for input_str in input_lines:
        if input_str.strip() == "":
            continue

        try:
            nums = ast.literal_eval(input_str)

            result = special_array(nums)

            print(result)
        except (ValueError, SyntaxError):
            print("Error: Invalid input")

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Basic Case with Distinct Values	Easy	Hidden	Success	0	0.0491 sec	11.3 KB
All Zeros	Easy	Hidden	Success	0	0.0499 sec	10.9 KB
Mixed Values with Multiple Valid x	Easy	Hidden	Success	0	0.0395 sec	11.2 KB
Case with Values in Descending Order	Easy	Hidden	Success	0	0.0553 sec	11.1 KB
Single Value Greater than Length	Easy	Hidden	Success	0	0.0465 sec	11.2 KB
Single Value Less than Length	Easy	Hidden	Success	0	0.0412 sec	11.1 KB
All Same Values Greater than	Easy	Hidden	Success	0	0.0403 sec	10.9 KB

Length						
Large List with No Valid x	Easy	Hidden	Success	0	0.0406 sec	11.2 KB
Decreasing Values with No Valid x	Easy	Hidden	Success	0	0.0494 sec	11.2 KB
Empty List	Easy	Hidden	Wrong Answer	0	0.0411 sec	11.2 KB
Pass/Fail Case	Easy	Hidden	Wrong Answer	0	0.0534 sec	11 KB

No comments.

4. What is the output of the following code snippet?

✓ Correct

Multiple Choice

Question description

```
lst = [1, 2, 3, 4]
lst[3] = 'banana'
print(lst)
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

○ [1, 2, &#39;banana&#39;, 4]

[● **[1, 2, 3, &#39;banana&#39;]**] ✓

○ Throws an error because all elements of a list must have the same data type.

○ Throws an error because index 3 does not exist within lst.

No comments.

5. What is the output of the following code snippet?

✓ Correct

Multiple Choice

Question description

```
def mystery_function(s, specific_digits):
    count = 0
    for char in s:
        if char in specific_digits:
            count += 1
    return count

result = mystery_function("There are 2 apples, 3 bananas, 5 cherries, and 7 dates.", "2378")
print(result)
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

- ○ 1
- ○ 2
- ● 3 ✓
- ○ 4

No comments.

6. What is the output?

Incorrect

Multiple Choice

Question description

```
def mystery_function(n):
    count = 0
    while count < n:
        count += 1
    return count

result = mystery_function(5)
print(result)
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

- [● 4]
- 5 ✓
- 6
- 10

No comments.

7. Find the bug!

✓ Correct

Coding

Question description

The provided code incorrectly implements the function `sort_by_parity`. Given a list of integers `nums`, `sort_by_parity` should return a new list that moves all of the even integers to the beginning of the list followed by all of the odd integers. Relative order of odd integers and even integers does not need to be maintained. Identify any bug(s) within the given implementation and correct the code so that it successfully passes the provided test cases.

```
def sort_by_parity(nums):
    evens = []
    odds = []
```

```
for num in nums:
    if num % 2 == 0:
        evens.append(num)
    if num % 2 == 1:
        odds.append(num)
return odds + evens
```

Candidate's Solution

Language used: **Python 3**

```
#!/bin/python3

import math
import os
import random
import re
import sys
import ast


#
# Complete the 'sort_by_parity' function below.
#
# The function is expected to return an INTEGER.
# The function accepts INTEGER_ARRAY nums as parameter.
#

def sort_by_parity(nums):
    evens = []
    odds = []
    for num in nums:
        if num % 2 == 0:
            evens.append(num)
        if num % 2 == 1:
            odds.append(num)
    return evens + odds

if __name__ == '__main__':
    input_str = sys.stdin.read().strip()
    input_list = ast.literal_eval(input_str)
    result = sort_by_parity(input_list)
    print(result)
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Pass/Fail Case	Easy	Hidden	Success	5	0.0417 sec	10.9 KB

No comments.