



Rajasekhar Reddy Kolagotla
Other

PDF generated at: 21 Jul 2025 01:44:40 UTC

[View this report on HackerRank](#)

Score

100% • 80 / 80
scored in TIP102: Unit 7 Version A (Standard) - Summer 2025 in 32 min 34 sec on 20 Jul 2025 18:10:09 PDT

Candidate Information

Email	rajasekhar1131997@gmail.com
Test	TIP102: Unit 7 Version A (Standard) - Summer 2025
Candidate Packet	View
Taken on	20 Jul 2025 18:10:09 PDT
Time taken	32 min 34 sec/ 90 min
Personal Member ID	126663
Email Address with CodePath	rajasekhar1131997@gmail.com
Github username with CodePath	Rajasekhar1131997
Invited by	CodePath

Suspicious Activity detected

Code similarity

Code similarity • 1 question

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 • 60 / 60

Status	No.	Question	Time Taken	Skill	Score	Code Quality
	1	Compute Power Function Coding	3 min 1 sec	-	20/20	-

	2	Bad Product Coding	6 min 50 sec	-	20/20	-
	3	First and Last Position of Element in Sorted Array Coding	15 min 46 sec	-	20/20	-

Multiple Choice + Debugging • 20 / 20

Status	No.	Question	Time Taken	Skill	Score	Code Quality
	4	What is the time complexity of mystery_function()? Multiple Choice	51 sec	-	5/5	-
	5	What is the output of the following code snippet? Multiple Choice	1 min 18 sec	-	5/5	-
	6	What is the output of the following code snippet? Multiple Choice	1 min 20 sec	-	5/5	-
	7	Debug this code Coding	3 min 12 sec	-	5/5	-

1. Compute Power Function

Correct

Coding

Question description

Given two integers, x and n , where n is non-negative, write a function `power()` that recursively computes and returns x^n .

Example 1:

Input: $x = 2, n = 3$

Output: 8

Explanation: $2^3 = 2 * 2 * 2 = 8$

Example 2:

Input: $x = 5, n = 0$

Output: 1

Explanation: $5^0 = 1$

Example 3:

Input: $x = 3, n = 2$

Output: 9

Explanation: $3^2 = 3 * 3 = 9$

Candidate's Solution

Language used: Python 3

```
1  #!/bin/python3
2
3  import math
4  import os
5  import random
6  import re
7  import sys
8  import ast
9
10
11
12 #
13 # Complete the 'power' function below.
14 #
15 # The function is expected to return an INTEGER.
16 # The function accepts following parameters:
```

```

17 # 1. INTEGER x
18 # 2. INTEGER n
19 #
20
21 def power(x, n):
22     # Write your code here
23     if n == 0:
24         return 1
25     return x * power(x, n-1)
26
27
28 if __name__ == '__main__':
29     outfile = open(os.environ['OUTPUT_PATH'], 'w')
30     input_data = sys.stdin.read().strip().splitlines()
31
32     results = []
33
34     for line in input_data:
35         x, n = eval(line)
36         result = power(x, n)
37         results.append(result)
38
39     for res in results:
40         outfile.write(str(res) + '\n')
41     outfile.close()
42

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Base case: Zero exponent	Easy	Hidden	Success	0	0.0301 sec	10.6 KB
Base case: Exponent of 1	Easy	Hidden	Success	0	0.0305 sec	10.6 KB
Small base and exponent	Easy	Hidden	Success	0	0.0371 sec	10.8 KB

Large base and small exponent	Easy	Hidden	Success	0	0.0289 sec	10.8 KB
Small base and large exponent	Easy	Hidden	Success	0	0.0293 sec	10.9 KB
Negative base with even exponent	Easy	Hidden	Success	0	0.0302 sec	10.9 KB
Negative base with odd exponent	Easy	Hidden	Success	0	0.0261 sec	10.8 KB
Negative base with zero exponent	Easy	Hidden	Success	0	0.027 sec	10.6 KB
Base is zero	Easy	Hidden	Success	0	0.0281 sec	10.8 KB
Base is zero and exponent is zero (0^0)	Easy	Hidden	Success	0	0.0309 sec	10.8 KB
Pass/Fail Case	Easy	Hidden	Success	20	0.0301 sec	10.8 KB

No comments.

2. Bad Product

Correct

Coding

Question description

You are given a list of characters `letters` that is sorted in **non-decreasing order**, and a character `target`. There are **at least two different** characters in `letters`.

Return *the smallest character in `letters` that is lexicographically greater (occurs later in the alphabet) than `target`*. If such a character does not exist, return the first character in `letters`.

Your solution must have $O(\log n)$ time complexity.

Example Input: `letters = ["c","f","j"], target = "a"`

Expected Output: `"c"`

Explanation: The smallest character that is lexicographically greater than 'a' in `letters` is 'c'.

Example Input: `letters = ["x","x","y","y"], target = "z"`

Expected Output: `"x"`

Explanation: There are no characters in `letters` that is lexicographically greater than 'z' so we return `letters[0]`.

Candidate's Solution

Language used: Python 3

```
1  #!/bin/python3
2
3  import math
4  import os
5  import random
6  import re
7  import sys
8  import ast
9
10
11
12 #
13 # Complete the 'next_greatest_letter' function below.
14 #
15 # The function is expected to return a STRING.
16 # The function accepts following parameters:
17 # 1. STRING_ARRAY letters
```

```

18 # 2. STRING target
19 #
20
21 def next_greatest_letter(letters, target):
22     # Write your code here
23     left = 0
24     right = len(letters) - 1
25     while left <= right:
26         mid = left + (right-left) // 2
27         if target >= letters[mid]:
28             left = mid + 1
29         else:
30             right = mid - 1
31     return letters[left % len(letters)]
32
33
34
35 if __name__ == '__main__':
36     outfile = open(os.environ['OUTPUT_PATH'], 'w')
37     input_data = sys.stdin.read().strip().splitlines()
38
39     results = []
40
41     for line in input_data:
42         letters, target = eval(line)
43         result = next_greatest_letter(letters, target)
44         results.append(result)
45
46     for res in results:
47         outfile.write(str(res) + '\n')
48     outfile.close()
49

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Basic Case	Easy	Hidden	Success	0	0.0328 sec	10.9 KB
No characters in letters that is	Easy	Hidden	Success	0	0.0398 sec	10.8 KB

lexicographically greater than target						
Target Matches a Letter in the List	Easy	Hidden	Success	0	0.0289 sec	10.9 KB
Target is Greater than All Letters	Easy	Hidden	Success	0	0.042 sec	10.9 KB
All Letters are the Same	Easy	Hidden	Success	0	0.0284 sec	10.9 KB
Single Letter in the List	Easy	Hidden	Success	0	0.0379 sec	10.9 KB
Target is Equal to the Last Letter	Easy	Hidden	Success	0	0.029 sec	10.9 KB
Target Falls Between Two Letters	Easy	Hidden	Success	0	0.0286 sec	10.9 KB
Large List with Repeated Characters	Easy	Hidden	Success	0	0.0289 sec	10.9 KB
Target is Smaller than All Letters in the List	Easy	Hidden	Success	0	0.0293 sec	10.9 KB
Wrap Around Case	Easy	Hidden	Success	0	0.0284 sec	10.9 KB

Pass/Fail Case

Easy

Hidden

Success

20

0.0323
sec

10.9 KB

 No comments.

3. First and Last Position of Element in Sorted Array

 Correct

Coding

Question description

Given an array of integers `nums` sorted in non-decreasing order, find the starting and ending position of a given target value.

If the target is not found in the array, return `[-1, -1]`.

You must write an algorithm with $O(\log n)$ runtime complexity.

Example 1:

Input: `nums = [5,7,7,8,8,10]`, `target = 8`

Output: `[3,4]`

Example 2:

Input: `nums = [5,7,7,8,8,10]`, `target = 6`

Output: `[-1,-1]`

Example 3:

Input: `nums = []`, `target = 0`

Output: `[-1,-1]`

Candidate's Solution

Language used: Python 3

```
1 #!/bin/python3
2
3 import math
```

```
4 import os
5 import random
6 import re
7 import sys
8 import ast
9
10
11
12 #
13 # Complete the 'search_range' function below.
14 #
15 # The function is expected to return an INTEGER.
16 # The function accepts following parameters:
17 # 1. INTEGER_ARRAY nums
18 # 2. INTEGER target
19 #
20
21 def search_range(nums, target):
22     # Write your code here
23     def find_first(nums, target):
24         left = 0
25         right = len(nums) - 1
26         first = -1
27         while left <= right:
28             mid = left + (right-left) // 2
29             if nums[mid] == target:
30                 first = mid
31                 right = mid - 1
32             elif target > nums[mid]:
33                 left = mid + 1
34             else:
35                 right = mid - 1
36         return first
37
38     def find_last(nums, target):
39         left = 0
40         right = len(nums) - 1
41         last = -1
42         while left <= right:
43             mid = left + (right-left) // 2
44             if nums[mid] == target:
45                 last = mid
46                 left = mid + 1
47             elif target > nums[mid]:
48                 left = mid + 1
49             else:
```

```

50             right = mid - 1
51         return last
52
53     return [find_first(nums,target), find_last(nums,target)]
54
55
56 if __name__ == '__main__':
57     outfile = open(os.environ['OUTPUT_PATH'], 'w')
58     input_data = sys.stdin.read().strip().splitlines()
59
60     results = []
61
62     for line in input_data:
63         nums, target = eval(line)
64         result = search_range(nums, target)
65         results.append(result)
66
67     for res in results:
68         outfile.write(str(res) + '\n')
69     outfile.close()

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Basic Case	Easy	Hidden	Success	0	0.0396 sec	10.8 KB
Basic Case	Easy	Hidden	Success	0	0.0281 sec	10.8 KB
Single Element Matching Target	Easy	Hidden	Success	0	0.0303 sec	10.9 KB
All Elements Match Target	Easy	Hidden	Success	0	0.0364 sec	10.8 KB
Target at the Beginning	Easy	Hidden	Success	0	0.0286 sec	10.8 KB

Target at the End	Easy	Hidden	Success	0	0.0324 sec	10.9 KB
Empty Array	Easy	Hidden	Success	0	0.0302 sec	10.8 KB
No Target Found	Easy	Hidden	Success	0	0.0313 sec	10.5 KB
Multiple Occurrences of Target	Easy	Hidden	Success	0	0.0321 sec	10.9 KB
Target Less than All Elements	Easy	Hidden	Success	0	0.0287 sec	10.9 KB
Target Greater than All Elements	Easy	Hidden	Success	0	0.0277 sec	10.9 KB
Pass/Fail Case	Easy	Hidden	Success	20	0.0272 sec	10.9 KB

🚫 No comments.

4. What is the time complexity of `mystery_function()`?

✅ Correct

Multiple Choice

Question description

What is the time complexity of `mystery_function()`?

```
def mystery_function(n):  
    if n == 0 or n == 1:  
        return 1  
    return n * factorial(n - 1)
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

☐ $O(1)$

☐ $O(\log n)$

☒ $O(n)$



☐ $O(n^2)$

 No comments.

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

 Correct

Multiple Choice

Question description

```
def mystery_function(arr):  
    if not arr:  
        return 0  
    return arr[0] + mystery_function(arr[1:])  
  
print(mystery_function([1, 2, 3, 4, 5]))  
print(mystery_function([10, 20, 30]))
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

☐

10

60

60

60

☐

15

70

70

70

☒

15

60

60

60



☐

15

50

50

50

 No comments.

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

 Correct

Multiple Choice

Question description

```
from collections import deque

def recursive_helper(queue):
    if not queue:
        return 0
    current = queue.popleft()
    return current + recursive_helper(queue)

def sum_with_queue(arr):
    queue = deque(arr)
    return recursive_helper(queue)

print(sum_with_queue([1, 2, 3, 4, 5]))
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

☐ 10

☐ 12

☐ 16

☒ 15



 No comments.

7. Debug this code

 Correct

Coding

Question description

The code provided below incorrectly implements the `binary_search()` function. Implemented correctly, `binary_search()` accepts a unique list of integers `nums` and an integer `target` and returns the index of `target` in the list. If `target` is not a value in the list, the function should return -1.

Identify any bug(s) within the given implementation and correct the code so that it successfully passes the provided test cases.


Candidate's Solution

Language used: Python 3

```
1  #!/bin/python3
2
3  import math
4  import os
5  import random
6  import re
7  import sys
8  import ast
9
10
11 def binary_search(nums, target):
12     left, right = 0, len(nums) - 1
13
14     while left <= right:
15         mid = left + (right - left) // 2
16         if nums[mid] == target:
17             return mid
18         elif target > nums[mid]:
19             left = mid + 1
20         else:
21             right = mid - 1
22
23     return -1
24
25 if __name__ == '__main__':
26     input_data = sys.stdin.read().strip()
27     input_list = ast.literal_eval(input_data)
```

```
27
28     nums = input_list[0]
29     target = input_list[1]
30
31     result = binary_search(nums, target)
32     print(result)
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Pass/Fail Case	Easy	Hidden	Success	5	0.0253 sec	10.9 KB

 No comments.