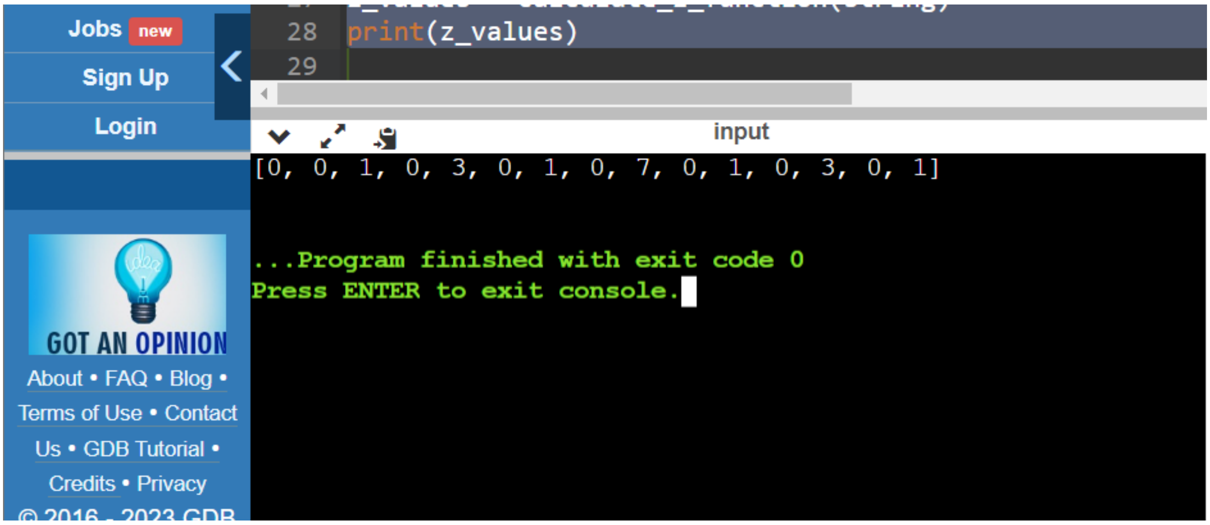
Name=Saransh Mittal
Rollno=2000290130145

Day-16:String-2

Problem Statement: Z-Function

```
def calculate_z_function(s):
    n = len(s)
    z = [0] * n
    left = right = 0

    for i in range(1, n):
        if i <= right:
            k = i - left
            if z[k] < right - i + 1:
                z[i] = z[k]
            else:
                left = i
                while right < n and s[right] == s[right - left]:
                    right += 1
                z[i] = right - left
                right -= 1

        else:
            left = right = i
            while right < n and s[right] == s[right - left]:
                right += 1
            z[i] = right - left
            right -= 1
    return z


string = "abacabadabacaba"

z_values = calculate_z_function(string)

print(z_values)
```



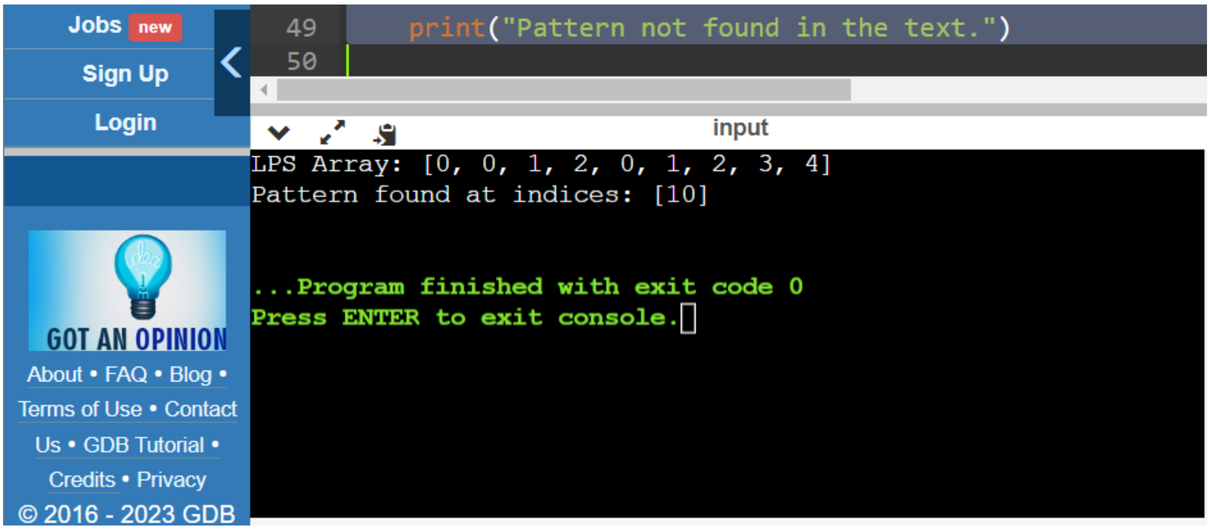
```
print(z_values)
```

```
[0, 0, 1, 0, 3, 0, 1, 0, 7, 0, 1, 0, 3, 0, 1]


...Program finished with exit code 0
Press ENTER to exit console.
```

Problem Statement: KMP algo/LPS(pi) array

```
def compute_lps(pattern):
    length = len(pattern)
    lps = [0] * length
    j = 0

    for i in range(1, length):
        while j > 0 and pattern[i] != pattern[j]:
            j = lps[j-1]

        if pattern[i] == pattern[j]:
            j += 1
            lps[i] = j
    return lps

def kmp_search(text, pattern):
    n = len(text)
    m = len(pattern)
    lps = compute_lps(pattern)

    i = j = 0
    indices = []

    while i < n:
        if pattern[j] == text[i]:
            i += 1
            j += 1

            if j == m:
                indices.append(i - j)
                j = lps[j-1]
        else:
            if j != 0:
                j = lps[j-1]
            else:
                i += 1

    return indices

text = "ABABDABACDABABCABAB"
pattern = "ABABCABAB"
lps_array = compute_lps(pattern)
print("LPS Array:", lps_array)
```

```
matches = kmp_search(text, pattern)

if matches:

   print("Pattern found at indices:", matches)

else:

   print("Pattern not found in the text.")
```

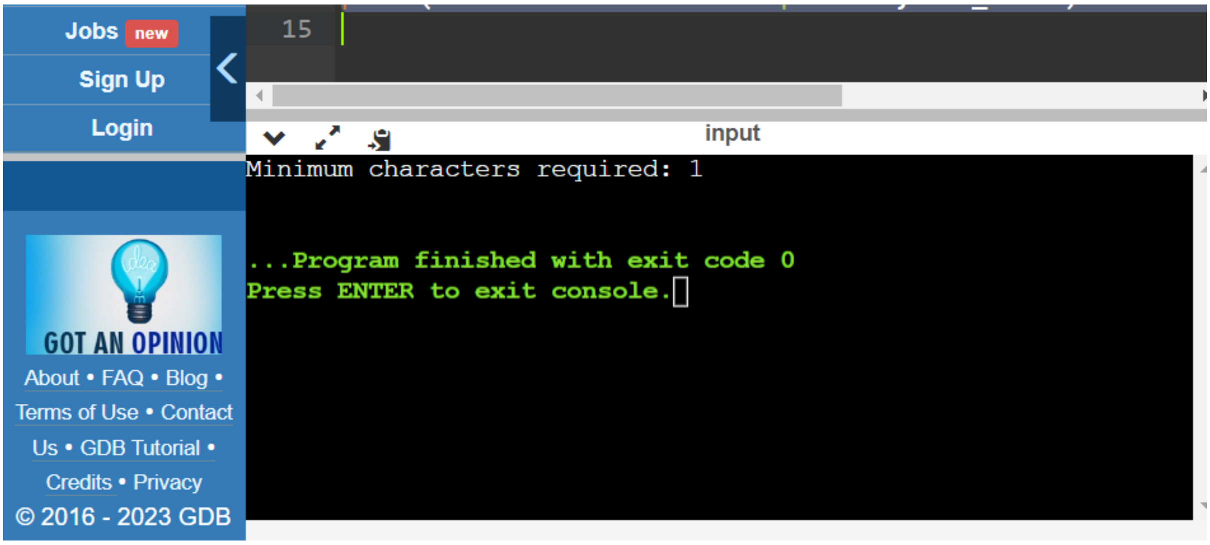


```
    print("Pattern not found in the text.")
```

```
LPS Array: [0, 0, 1, 2, 0, 1, 2, 3, 4]
Pattern found at indices: [10]


...Program finished with exit code 0
Press ENTER to exit console.
```

Problem Statement: Minimum Characters required to make a String Palindromic

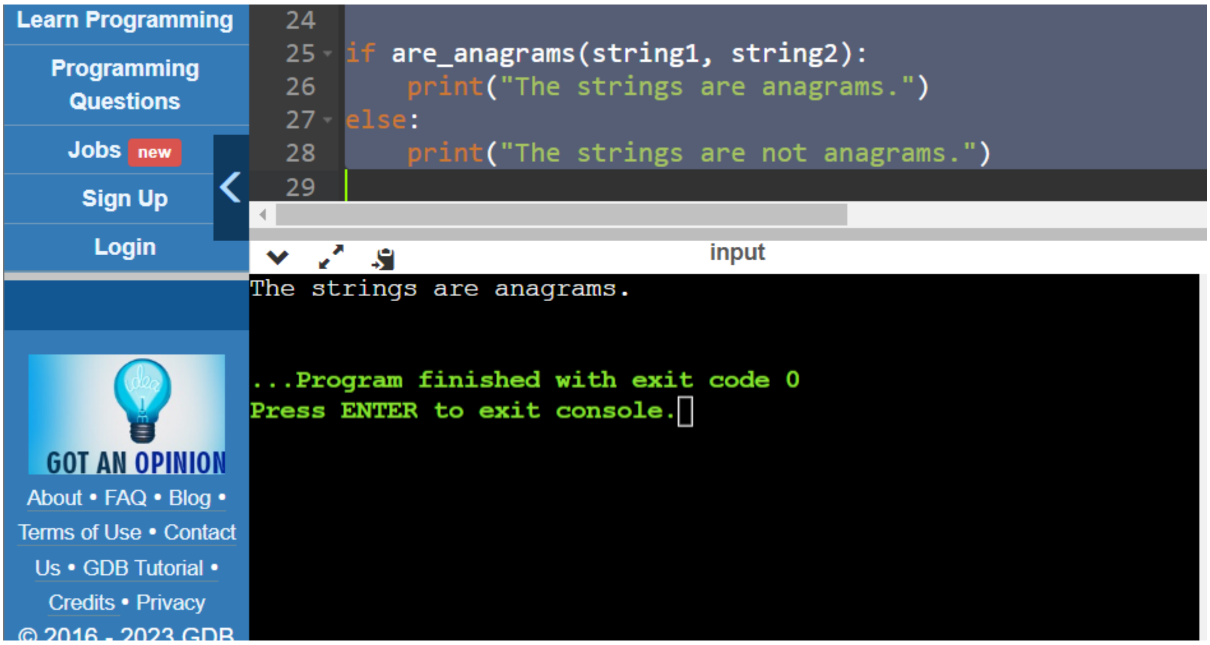
```
def min_chars_to_palindrome(string):

   if string == string[::-1]:

      return 0



   n = len(string)

   for i in range(n, 0, -1):

      if string[:i] == string[i - 1::-1]:

         return n - i

   return n - 1




input_string = "ab"

min_chars = min_chars_to_palindrome(input_string)

print("Minimum characters required:", min_chars)
```



```
Minimum characters required: 1


...Program finished with exit code 0
Press ENTER to exit console.
```

Problem Statement: Check for Anagrams

```
def are_anagrams(str1, str2):
    str1 = str1.lower().replace(" ", "")
    str2 = str2.lower().replace(" ", "")

    if len(str1) != len(str2):
        return False

    char_count1 = {}
    char_count2 = {}

    for char in str1:
        char_count1[char] = char_count1.get(char, 0) + 1
    for char in str2:
        char_count2[char] = char_count2.get(char, 0) + 1

    if char_count1 == char_count2:
        return True
    else:
        return False

string1 = "listen"
string2 = "silent"

if are_anagrams(string1, string2):
    print("The strings are anagrams.")
else:
    print("The strings are not anagrams.")
```



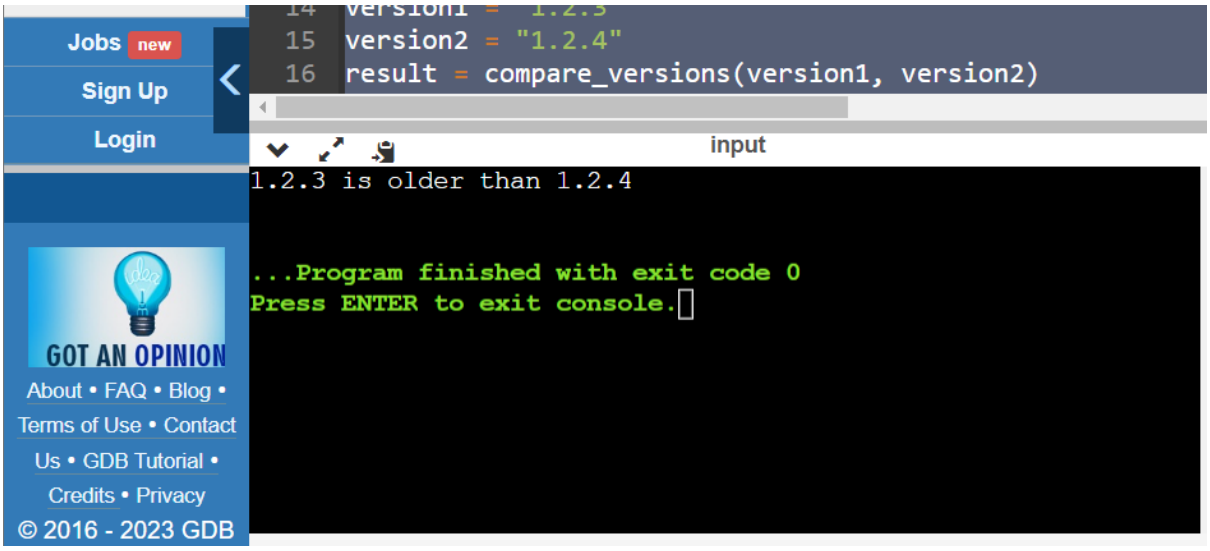
```
if are_anagrams(string1, string2):
    print("The strings are anagrams.")
else:
    print("The strings are not anagrams.")
```

```
The strings are anagrams.


...Program finished with exit code 0
Press ENTER to exit console.
```

Problem Statement: Compare version numbers

```
def compare_versions(version1, version2):
    v1 = version1.split('.')
    v2 = version2.split('.')
    for i in range(max(len(v1), len(v2))):
        num1 = int(v1[i]) if i < len(v1) else 0
        num2 = int(v2[i]) if i < len(v2) else 0

        if num1 < num2:
            return -1
        elif num1 > num2:
            return 1
    return 0

version1 = "1.2.3"
version2 = "1.2.4"
result = compare_versions(version1, version2)

if result < 0:
    print(f"{version1} is older than {version2}")
elif result > 0:
    print(f"{version1} is newer than {version2}")
```

```
else:
    print(f"{version1} and {version2} are the same version")
```

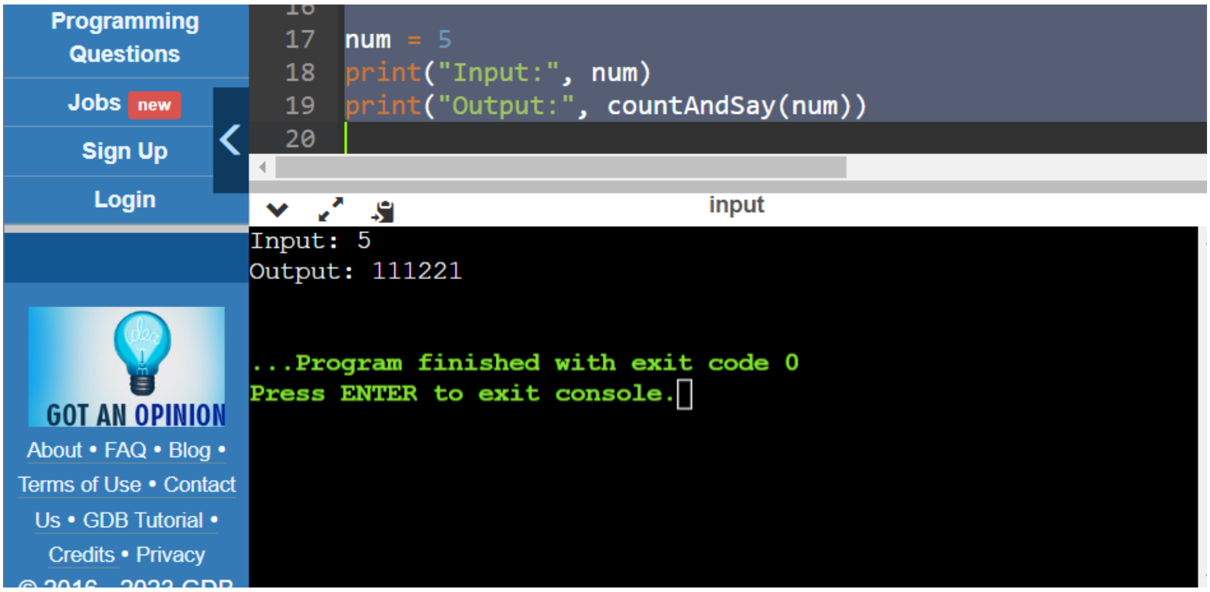

Jobs new
Sign Up
Login
GOT AN OPINION
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2023 GDB
input

```
version1 = "1.2.3"
version2 = "1.2.4"
result = compare_versions(version1, version2)
```

```
1.2.3 is older than 1.2.4


...Program finished with exit code 0
Press ENTER to exit console.
```

Problem Statement : Count and say

```
def countAndSay(n):

if n == 1:

    return "1"



prev_seq = countAndSay(n - 1)

count = 1

result = ""

for i in range(len(prev_seq)):

    if i + 1 < len(prev_seq) and prev_seq[i] == prev_seq[i + 1]:

        count += 1

    else:

        result += str(count) + prev_seq[i]
        count = 1
    return result



num = 5
print("Input:", num)
print("Output:", countAndSay(num))
```



Programming Questions
Jobs new
Sign Up
Login
GOT AN OPINION
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
input

```
num = 5
print("Input:", num)
print("Output:", countAndSay(num))
```

```
Input: 5
Output: 111221


...Program finished with exit code 0
Press ENTER to exit console.
```