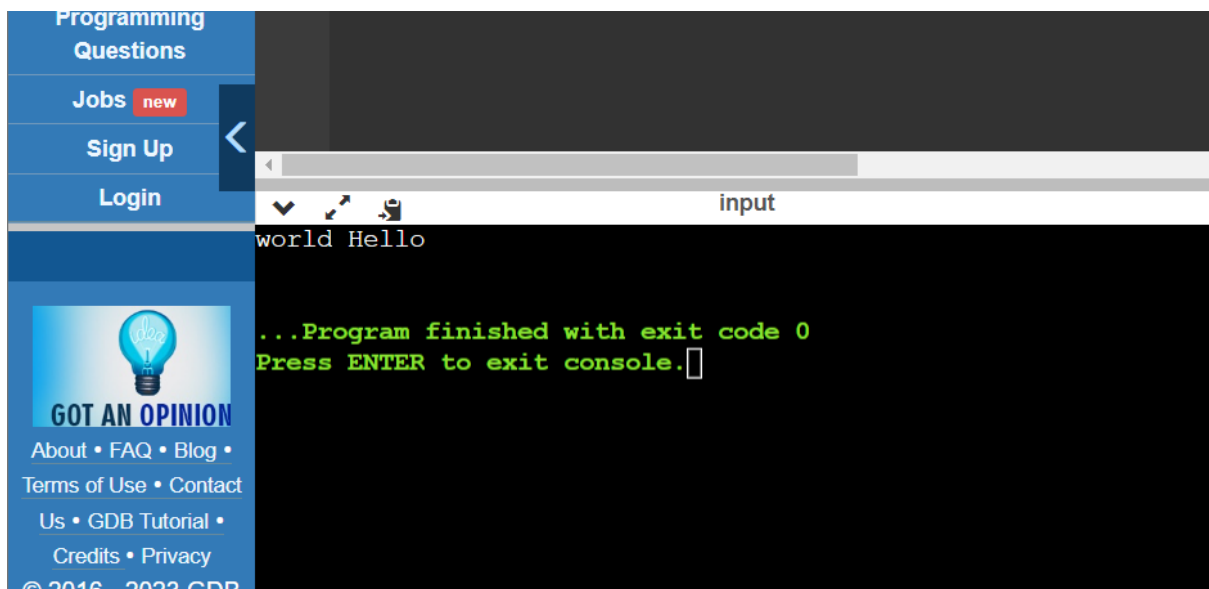


Day-15:String-1

Problem Statement: Given a string *s*, reverse the words of the string.

```
def reverse_words(s):  
    words = s.split(' ')  
    reversed_words = words[::-1]  
    reversed_string = ' '.join(reversed_words)  
    return reversed_string  
  
string = "Hello world"  
reversed_string = reverse_words(string)  
print(reversed_string)
```



Problem Statement: Longest Palindromic Substring

```
def expandAroundCenter(s, left, right):  
    while left >= 0 and right < len(s) and s[left] == s[right]:  
        left -= 1  
        right += 1  
    return right - left - 1  
  
def longestPalindrome(s):
```

```

start = 0
maxLen = 0

for i in range(len(s)):
    len1 = expandAroundCenter(s, i, i)
    len2 = expandAroundCenter(s, i, i + 1)

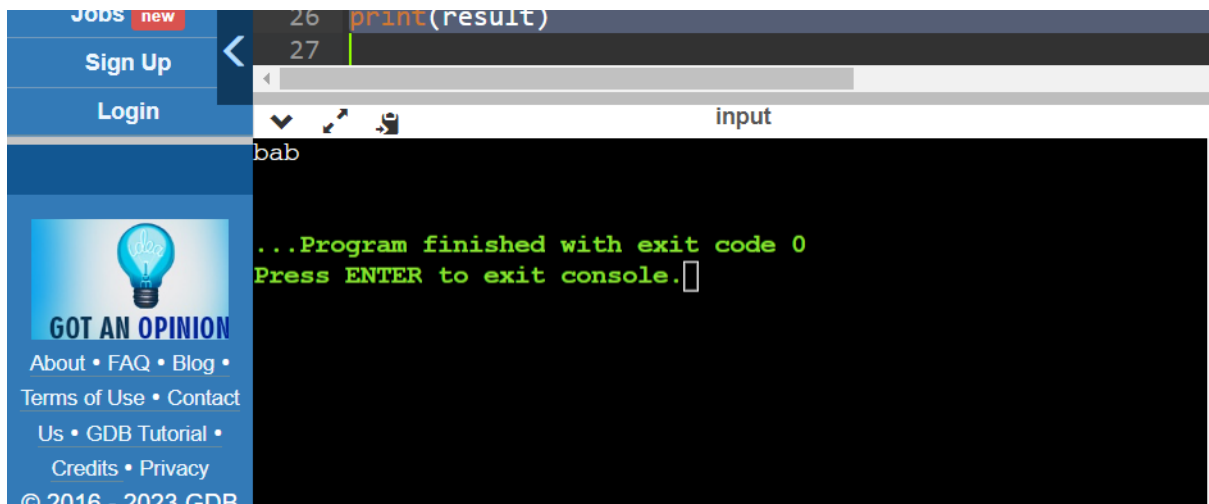
    if len1 > maxLen:
        maxLen = len1
        start = i - (len1 - 1) // 2

    if len2 > maxLen:
        maxLen = len2
        start = i - len2 // 2 + 1

return s[start:start + maxLen]

s = "babad"
result = longestPalindrome(s)
print(result)

```



```

26 print(result)
27
input
bab
...Program finished with exit code 0
Press ENTER to exit console.

```

Problem Statement: Roman Number to Integer and vice versa

```
def roman_to_integer(roman):
```

```
roman_dict = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}
```

```
result = 0
```

```
prev_value = 0
```

```
for char in reversed(roman):
```

```
    value = roman_dict[char]
```

```
    if value >= prev_value:
```

```
        result += value
```

```
    else:
```

```
        result -= value
```

```
    prev_value = value
```

```
return result
```

```
def integer_to_roman(num):
```

```
    integer_dict = {1000: 'M', 900: 'CM', 500: 'D', 400: 'CD', 100: 'C', 90: 'XC', 50: 'L', 40: 'XL',
```

```
                    10: 'X', 9: 'IX', 5: 'V', 4: 'IV', 1: 'I'}
```

```
    result = ""
```

```
    for value, symbol in integer_dict.items():
```

```
        while num >= value:
```

```
            result += symbol
```

```
            num -= value
```

```
    return result
```

```
roman_number = "II"
```

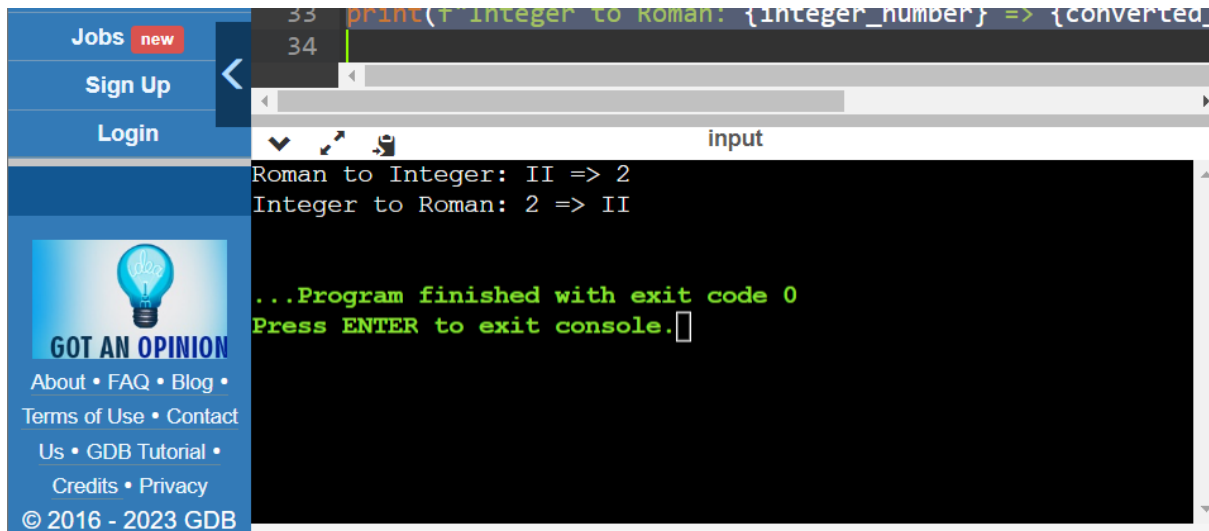
```
integer_number = 2
```

```
converted_integer = roman_to_integer(roman_number)
```

```
converted_roman = integer_to_roman(integer_number)
```

```
print(f"Roman to Integer: {roman_number} => {converted_integer}")
```

```
print(f"Integer to Roman: {integer_number} => {converted_roman}")
```



The screenshot shows a web application interface. On the left is a blue sidebar menu with the following items: 'Jobs' with a 'new' badge, 'Sign Up', 'Login', and a section titled 'GOT AN OPINION' featuring a lightbulb icon and links for 'About', 'FAQ', 'Blog', 'Terms of Use', 'Contact Us', 'GDB Tutorial', 'Credits', and 'Privacy'. At the bottom of the sidebar is the copyright notice '© 2016 - 2023 GDB'. The main content area on the right displays a terminal window with a dark background. The terminal shows the following output: 'Roman to Integer: II => 2', 'Integer to Roman: 2 => II', and a green message '...Program finished with exit code 0' followed by 'Press ENTER to exit console.' with a cursor. Above the terminal, there is a header bar with a search icon and the word 'input'.

Problem Statement: Implement ATOI/STRSTR

```
def atoi(s):
```

```
    s = s.strip()
```

```
    if not s:
```

```
        return 0
```

```
    sign = -1 if s[0] == '-' else 1
```

```
    if s[0] in ('-', '+'):
```

```
        s = s[1:]
```

```
    result = 0
```

```
    for char in s:
```

```
if not char.isdigit():  
    break  
  
result = result * 10 + int(char)
```

```
return sign * result
```

```
def strstr(haystack, needle):
```

```
    if not needle:  
        return 0
```

```
    for i in range(len(haystack) - len(needle) + 1):
```

```
        j = 0
```

```
        while j < len(needle) and haystack[i+j] == needle[j]:
```

```
            j += 1
```

```
        if j == len(needle):
```

```
            return i
```

```
    return -1
```

```
str1 = "12345"
```

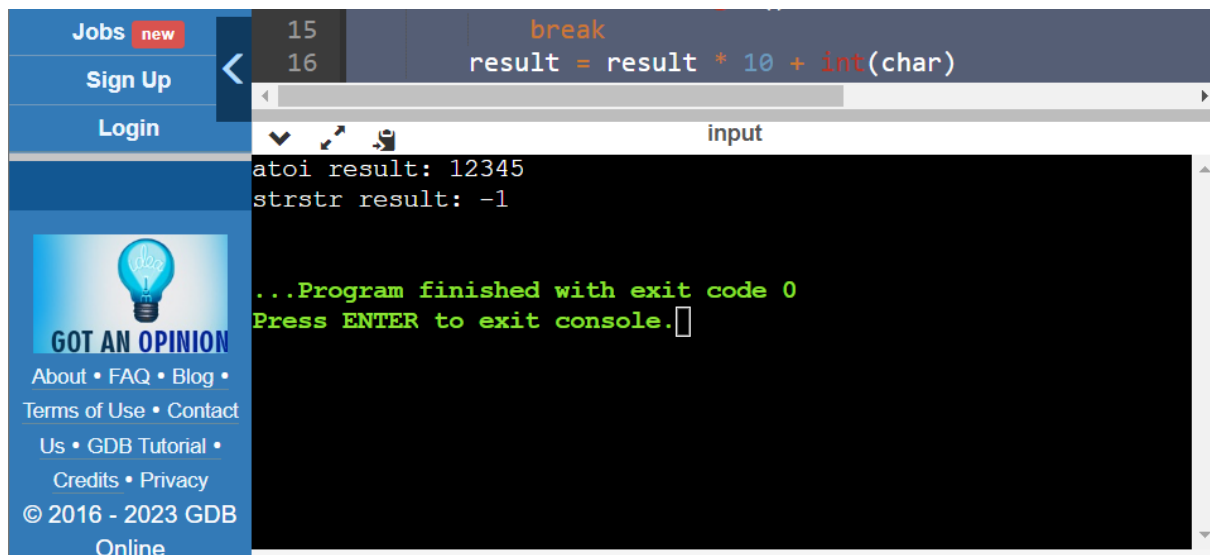
```
str2 = "567"
```

```
result1 = atoi(str1)
```

```
result2 = strstr(str1, str2)
```

```
print("atoi result:", result1)
```

```
print("strstr result:", result2)
```



The screenshot shows a web-based IDE interface. On the left is a sidebar with navigation links: 'Jobs' (with a 'new' badge), 'Sign Up', 'Login', and a 'GOT AN OPINION' section with links to 'About', 'FAQ', 'Blog', 'Terms of Use', 'Contact', 'Us', 'GDB Tutorial', 'Credits', and 'Privacy'. The main area displays C code in a dark-themed editor. Line 15 contains 'break' and line 16 contains 'result = result * 10 + int(char)'. Below the code editor, the output window shows 'atoi result: 12345' and 'strstr result: -1'. At the bottom of the output window, it says '...Program finished with exit code 0' and 'Press ENTER to exit console.'.

Problem Statement: Longest Common Prefix

```
def longest_common_prefix(strs):
```

```
    if not strs:
```

```
        return ""
```

```
    min_len = min(len(s) for s in strs)
```

```
    low, high = 0, min_len - 1
```

```
    while low <= high:
```

```
        mid = (low + high) // 2
```

```
        prefix = strs[0][:mid + 1]
```

```
        if all(s.startswith(prefix) for s in strs):
```

```
            low = mid + 1
```

else:

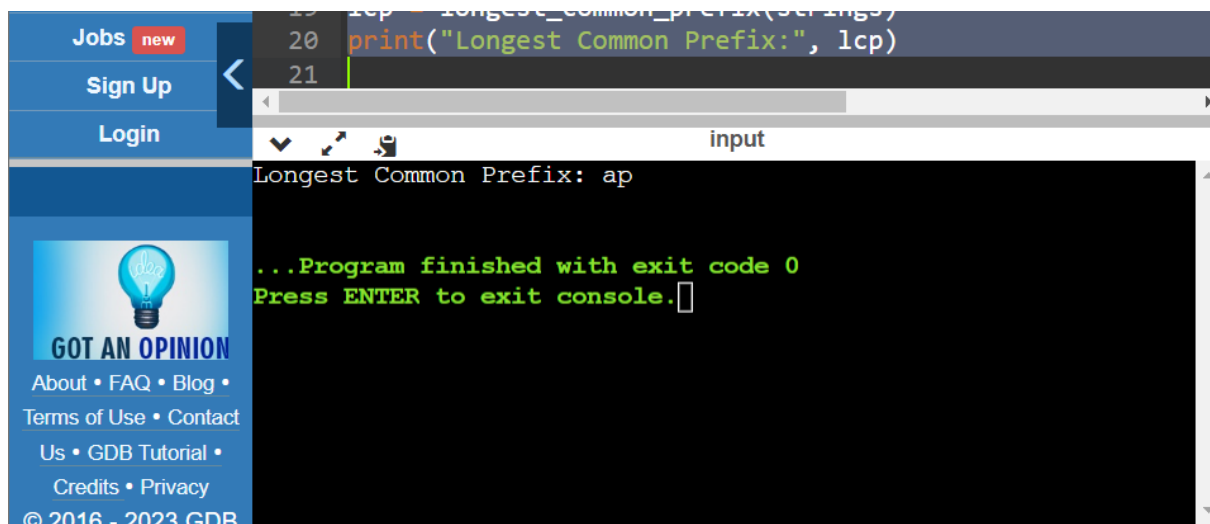
high = mid - 1

return strs[0][:high + 1]

strings = ["apple", "ape", "april"]

lcp = longest_common_prefix(strings)

print("Longest Common Prefix:", lcp)



The screenshot shows a web browser interface. On the left is a sidebar with a blue header containing 'Jobs new', 'Sign Up', and 'Login'. Below this is a section titled 'GOT AN OPINION' with a lightbulb icon and links for 'About', 'FAQ', 'Blog', 'Terms of Use', 'Contact', 'Us', 'GDB Tutorial', 'Credits', and 'Privacy'. The footer of the sidebar says '© 2016 - 2023 GDB'. The main content area on the right shows a terminal window with a dark background. The terminal output displays the result of a Python script: 'Longest Common Prefix: ap'. Above this, the terminal shows the execution of the script, including the line 'print("Longest Common Prefix:", lcp)'. The terminal also shows the message '...Program finished with exit code 0' and 'Press ENTER to exit console.'.

Problem Statement :Rabin karp

```
def rabin_karp(text, pattern):
```

```
    indices = []
```

```
    if not text or not pattern or len(pattern) > len(text):
```

```
        return indices
```

```
    prime = 101
```

```
d = 256
```

```
n = len(text)
```

```
m = len(pattern)
```

```
pattern_hash = 0
```

```
text_hash = 0
```

```
h = pow(d, m-1, prime)
```

```
for i in range(m):
```

```
    pattern_hash = (d * pattern_hash + ord(pattern[i])) % prime
```

```
    text_hash = (d * text_hash + ord(text[i])) % prime
```

```
for i in range(n - m + 1):
```

```
    if pattern_hash == text_hash:
```

```
        match = True
```

```
        for j in range(m):
```

```
            if text[i+j] != pattern[j]:
```

```
                match = False
```

```
                break
```

```
        if match:
```

```
            indices.append(i)
```



```
if i < n - m:
```

```
    text_hash = (d * (text_hash - ord(text[i]) * h) + ord(text[i + m])) % prime
```

```
    if text_hash < 0:
```

```
        text_hash += prime
```

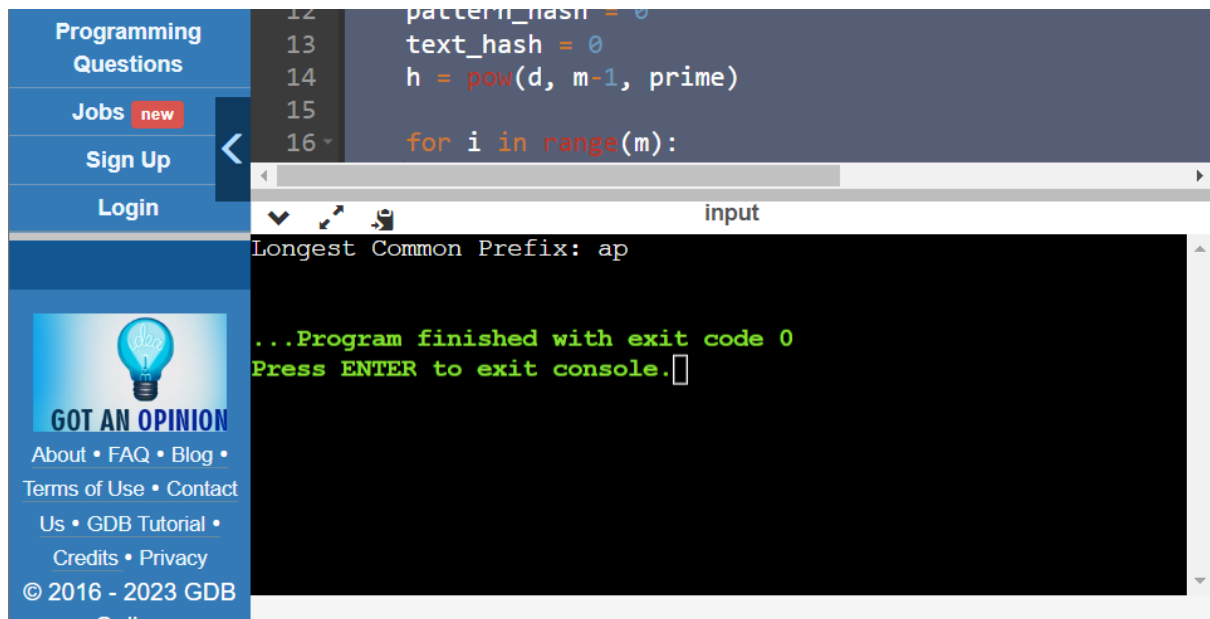
```
return indices
```

```
text = "ABABDABACDABABCABAB"
```

```
pattern = "ABABCABAB"
```

```
result = rabin_karp(text, pattern)
```

```
print("Pattern found at indices:", result)
```



```
12     pattern_hash = 0
13     text_hash = 0
14     h = pow(d, m-1, prime)
15
16     for i in range(m):
```

input

Longest Common Prefix: ap

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

GOT AN OPINION
About • FAQ • Blog •
Terms of Use • Contact
Us • GDB Tutorial •
Credits • Privacy
© 2016 - 2023 GDB