Run | Debug | Stop | Share | Save | {} Beautify | Language Python 3

main.py

```python
def min_operations(arr1, arr2):
    n, m = len(arr1), len(arr2)
    dp = [[float('inf')] * (m + 1) for _ in range(n + 1)]
    dp[0][0] = 0

    for i in range(1, n + 1):
        for j in range(1, m + 1):
            if arr1[i - 1] > dp[i - 1][j - 1]:
                dp[i][j] = min(dp[i][j], dp[i - 1][j - 1])
            if arr2[j - 1] > dp[i - 1][j - 1]:
                dp[i][j] = min(dp[i][j], dp[i - 1][j - 1] + 1)

    result = min(dp[n])
    return result if result != float('inf') else -1

arr1 = [1, 5, 3, 6, 7]
arr2 = [1, 3, 2, 4]
print(min_operations(arr1, arr2))

```

input

```
-1


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

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
def min_repeats(a, b):
    if b in a:
        return 1
    for i in range(1, len(b)):
        if b.startswith(a[i:]):
            return 1 + min_repeats(a, b[i:])
    return -1

a = "abcd"
b = "cdabcdab"
output = min_repeats(a, b)
print(output)

```

input

```
3


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

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
def missing_number(nums):
    n = len(nums)
    total_sum = n * (n + 1) // 2
    actual_sum = sum(nums)
    return total_sum - actual_sum

nums = [3, 0, 1]
print(missing_number(nums))

```

input

```
2


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

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
from collections import deque

def updateMatrix(mat):
    rows, cols = len(mat), len(mat[0])
    queue = deque()
    for i in range(rows):
        for j in range(cols):
            if mat[i][j] == 0:
                queue.append((i, j))
            else:
                mat[i][j] = float('inf')
    directions = [(0, 1), (0, -1), (1, 0), (-1, 0)]
    while queue:
        cell = queue.popleft()
        for d in directions:
            new_i, new_j = cell[0] + d[0], cell[1] + d[1]
            if 0 <= new_i < rows and 0 <= new_j < cols and mat[new_i][new_j] > mat[cell[0]][cell[1]] + 1:
                mat[new_i][new_j] = mat[cell[0]][cell[1]] + 1
                queue.append((new_i, new_j))
    return mat

mat1 = [[0, 0, 0], [0, 1, 0], [0, 0, 0]]
mat2 = [[0, 0, 0], [0, 1, 0], [1, 1, 1]]

print(updateMatrix(mat1))
print(updateMatrix(mat2))

```

input

```
[[0, 0, 0], [0, 1, 0], [0, 0, 0]]
[[0, 0, 0], [0, 1, 0], [1, 2, 1]]


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

main.py

```python
def stringMatching(words):
    return [word for word in words if any(other_word.find(word) != -1 for other_word in words if word != other_word)]

words = ["mass", "as", "hero", "superhero"]
print(stringMatching(words))

```

Input

```
['as', 'hero']


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

main.py

```python
def minOperations(nums):
    stack = []
    for num in nums:
        if stack and num < stack[-1]:
            stack.pop()
        else:
            stack.append(num)
    return len(stack)

nums = [1, 2, 3, 4, 5, 6]
print(minOperations(nums))

```

Input

```
6


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

Run Debug Stop Share Save {} Beautify

Language Python 3

main.py

```python
def isPrefixOfWord(sentence, searchWord):
    words = sentence.split()
    for i, word in enumerate(words, 1):
        if word.startswith(searchWord):
            return i
    return -1

sentence = "i love eating burger"
searchWord = "love"
print(isPrefixOfWord(sentence, searchWord))

```

input

```
2


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

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
def find_nth_digit(n):
    length = 1
    count = 9
    start = 1

    while n > length * count:
        n -= length * count
        length += 1
        count *= 10
        start *= 10

    start += (n - 1) // length
    s = str(start)
    return int(s[(n - 1) % length])

n = 15
print(find_nth_digit(n))
```

input

```
2


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

Run | Debug | Stop | Share | Save | {} Beautify | Language Python 3

main.py

```python
def transpose(matrix):
    return [list(row) for row in zip(*matrix)]

matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print(transpose(matrix))

```

input

```
[[1, 4, 7], [2, 5, 8], [3, 6, 9]]


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

main.py

```python
MOD = 10**9 + 7

def count_good_strings(s1, s2, evil):
    n = len(s1)
    m = len(evil)

    def next_state(state, char):
        while state > 0 and evil[state] != char:
            state = kmp[state - 1]
        if evil[state] == char:
            state += 1
        return state

    def dp(idx, state, tight1, tight2):
        if state == m:
            return 0
        if idx == n:
            return 1
        if memo[idx][state][tight1][tight2] != -1:
            return memo[idx][state][tight1][tight2]

        low = s1[idx] if tight1 else 'a'
        high = s2[idx] if tight2 else 'z'

        result = 0
        for c in range(ord(low), ord(high) + 1):
            new_tight1 = tight1 and (c == ord(low))
            new_tight2 = tight2 and (c == ord(high))
            new_state = next_state(state, chr(c))
```

input

```
0


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