e_python_compiler

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

main.py

```python
def removeElement(nums, val):
    nums[:] = [x for x in nums if x != val]
    return len(nums)

# Example
nums = [3, 2, 2, 3]
val = 3
print(removeElement(nums, val))
```

input

```
2


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

```python
def solveSudoku(board):
    def isValid(x, y, c):
        return all(c != board[i][y] for i in range(9)) and \
               all(c != board[x][j] for j in range(9)) and \
               all(c != board[x // 3 * 3 + i][y // 3 * 3 + j] for i in range(3) for j in range(3))
    def solve():
        for i in range(9):
            for j in range(9):
                if board[i][j] == '.':
                    for c in '123456789':
                        if isValid(i, j, c):
                            board[i][j] = c
                            if solve():
                                return True
                            board[i][j] = '.'
                    return False
        return True
    solve()
board = [
    ["5", "3", ".", ".", "7", ".", ".", ".", "."],
    ["6", ".", ".", "1", "9", "5", ".", ".", "."],
    [".", "9", "8", ".", ".", ".", ".", "6", "."],
    ["8", ".", ".", ".", "6", ".", ".", ".", "3"],
    ["4", ".", ".", "8", ".", "3", ".", ".", "1"],
    ["7", ".", ".", ".", "2", ".", ".", ".", "6"],
    [".", "6", ".", ".", ".", ".", "2", "8", "."],
    [".", ".", ".", "4", "1", "9", ".", ".", "5"],
    [".", ".", ".", ".", "8", ".", ".", "7", "9"]
]
solveSudoku(board)
for row in board:
    print(row)
```

input

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

input

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

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

main.py

```python
def countAndSay(n):
    if n == 1:
        return "1"
    prev = countAndSay(n - 1)
    result = ""
    count = 1
    for i in range(1, len(prev)):
        if prev[i] == prev[i - 1]:
            count += 1
        else:
            result += str(count) + prev[i - 1]
            count = 1
    result += str(count) + prev[-1]
    return result

# Example
print(countAndSay(4))
```

input

```
1211


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

main.py

```python
def combinationSum(candidates, target):
    res = []

    def backtrack(start, path, target):
        if target == 0:
            res.append(path)
            return
        if target < 0:
            return
        for i in range(start, len(candidates)):
            backtrack(i, path + [candidates[i]], target - candidates[i])

    candidates.sort()
    backtrack(0, [], target)
    return res

# Example
candidates = [2, 3, 6, 7]
target = 7
print(combinationSum(candidates, target))
```

input

```
[[2, 2, 3], [7]]


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

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

main.py

```python
def combinationSum2(candidates, target):
    res = []

    def backtrack(start, path, target):
        if target == 0:
            res.append(path)
            return
        if target < 0:
            return
        for i in range(start, len(candidates)):
            if i > start and candidates[i] == candidates[i - 1]:
                continue
            backtrack(i + 1, path + [candidates[i]], target - candidates[i])

    candidates.sort()
    backtrack(0, [], target)
    return res

# Example
candidates = [10, 1, 2, 7, 6, 1, 5]
target = 8
print(combinationSum2(candidates, target))
```

input

```
[[1, 1, 6], [1, 2, 5], [1, 7], [2, 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 permuteUnique(nums):
    res = []

    def backtrack(nums, path):
        if not nums:
            res.append(path)
            return
        for i in range(len(nums)):
            if i > 0 and nums[i] == nums[i - 1]:
                continue
            backtrack(nums[:i] + nums[i + 1:], path + [nums[i]])

    nums.sort()
    backtrack(nums, [])
    return res

# Example
nums = [1, 1, 2]
print(permuteUnique(nums))
```

input

```
[[1, 1, 2], [1, 2, 1], [2, 1, 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 maxSubArray(nums):
    max_sum = current_sum = nums[0]
    for num in nums[1:]:
        current_sum = max(num, current_sum + num)
        max_sum = max(max_sum, current_sum)
    return max_sum

# Example
nums = [-2, 1, -3, 4, -1, 2, 1, -5, 4]
print(maxSubArray(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 lengthOfLastWord(s):
    s = s.strip()
    return len(s.split()[-1]) if s else 0

# Example
s = "Hello World"
print(lengthOfLastWord(s))
```

input

```
5


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

main.py

```python
import math

def getPermutation(n, k):
    nums = [str(i) for i in range(1, n + 1)]
    result = ""
    k -= 1
    while n > 0:
        n -= 1
        index, k = divmod(k, math.factorial(n))
        result += nums.pop(index)
    return result

# Example
n = 3
k = 3
print(getPermutation(n, k))
```

input

```
213


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

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

main.py

```python
def isValidSudoku(board):
    seen = set()
    for i in range(9):
        for j in range(9):
            if board[i][j] != '.':
                current_number = board[i][j]
                if (current_number, i) in seen or (j, current_number) in seen or (i // 3, j // 3, current_number) in seen:
                    return False
                seen.add((current_number, i))
                seen.add((j, current_number))
                seen.add((i // 3, j // 3, current_number))
    return True
board = [
    ["5", "3", ".", ".", "7", ".", ".", ".", "."],
    ["6", ".", ".", "1", "9", "5", ".", ".", "."],
    [".", "9", "8", ".", ".", ".", ".", "6", "."],
    ["8", ".", ".", ".", "6", ".", ".", ".", "3"],
    ["4", ".", ".", "8", ".", "3", ".", ".", "1"],
    ["7", ".", ".", ".", "2", ".", ".", ".", "6"],
    [".", "6", ".", ".", ".", ".", "2", "8", "."],
    [".", ".", ".", "4", "1", "9", ".", ".", "5"],
    [".", ".", ".", ".", "8", ".", ".", "7", "9"]
]
print(isValidSudoku(board))

```

input

```
True


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