main.py

```python
def remove(nums, val):
    i = 0
    for j in range(len(nums)):
        if nums[j] != val:
            nums[i] = nums[j]
            i += 1
    return i
nums = [3, 2, 2, 3]
val = 3
print(remove(nums, val))
```

input

```
2


...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

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

input

```
213


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

main.py F9

```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

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.
```

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

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.
```

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
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.
```

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

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.
```

main.py

```python
def solvesudoku(board):
    def is_valid(board, row, col, num):
        sub_box_row = (row // 3) * 3
        sub_box_col = (col // 3) * 3
        for i in range(9):
            if board[row][i] == num or board[i][col] == num:
                return False
            if board[sub_box_row + i // 3][sub_box_col + i % 3] == num:
                return False
        return True

    def back(board):
        for row in range(9):
            for col in range(9):
                if board[row][col] == '.':
                    for num in map(str, range(1, 10)):
                        if is_valid(board, row, col, num):
                            board[row][col] = num
                            if back(board):
                                return True
                            board[row][col] = '.'
                    return False
        return True

    back(board)
sudoku = [
    ['5', '3', '.', '.', '7', '.', '.', '.', '.'],
    ['6', '.', '.', '1', '9', '5', '.', '.', '.'],
```

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']


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

```python
def lengthOfLastWord(s):
    s = s.strip()
    return len(s.split()[-1]) if s else 0

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

input

```
5


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