```
def superReducedString(s):
    # Write your code here
    ew = False
    while ew == False:
        s = re.sub(r"(.)\1",r"" , s)
        ew = False if re.search(r"(.)\1", s) else True
    return s if s != "" else "Empty String"

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    s = input()

    result = superReducedString(s)

    fptr.write(result + '\n')

    fptr.close()
```

Super reduced string

```
def countingValleys(steps, path):
    # Write your code here
    alt, val = 0, 0
    enterVal = False
    for i in path:
        if i == 'D':
            alt -= 1
            if not enterVal and alt < 0:
                enterVal = True
        elif i == 'U':
            alt += 1
            if enterVal and alt == 0:
                val += 1
                enterVal = False
    return val

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    steps = int(input().strip())

    path = input()

    result = countingValleys(steps, path)

    fptr.write(str(result) + '\n')
```

```
    fptr.close()
```

Counting valleys

```
def jumpingOnClouds(c):
    # Write your code here
    n=len(c)
    steps=0
    i=0
    while(i<n-2):
        i= i+2 if c[i+2]==0 else i+1
        steps= steps+1
    if i==n-2:
         steps+=1
    return steps

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    n = int(input().strip())

    c = list(map(int, input().rstrip().split()))

    result = jumpingOnClouds(c)

    fptr.write(str(result) + '\n')

    fptr.close()
```

Jumping on clouds

```
def camelcase(s):
    # Write your code here
    words = 1
    print("C"<"Z")
    for i in s:
        if(i>="A" and i<="Z"):
```

```
            words +=1
    return words

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    s = input()

    result = camelcase(s)

    fptr.write(str(result) + '\n')

    fptr.close()
```

Camel case

```
def minimumNumber(n, password):
    # Return the minimum number of characters to make the password strong
    count = 0
    if not(re.search(r'[A-Z]+',password)): count += 1
    if not(re.search(r'[a-z]+',password)): count += 1
    if not(re.search(r'[0-9]+',password)): count += 1
    if not(re.search(r'[!@#$%^&*()+\-]+',password)): count += 1
    if n <= 3 or 6-n >= count: count = 6 - n
    return count

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    n = int(input().strip())

    password = input()

    answer = minimumNumber(n, password)

    fptr.write(str(answer) + '\n')

    fptr.close()
```

Strong password

```
def alternate(s):
    # Write your code here
    letters = list(set(list(s)))
    words = {}
```

```
    for i in range(len(letters)):
        for j in range(i+1,len(letters)):
            w = ''
            for l in s:
                if l == letters[i] or l == letters[j]:
                    if w == '' or l != w[-1]:
                        w += l
                    else:
                        w = ''
                        break
            if w != '' and len(w) > 1:
                words[w] = len(w)

    if len(words) == 0:
        return 0
    return max(words.values())

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    l = int(input().strip())

    s = input()

    result = alternate(s)

    fptr.write(str(result) + '\n')

    fptr.close()
```

Two characters

```
def caesarCipher(s, k):
    # Write your code here
    import string
    chr_l = string.ascii_lowercase
    chr_u = string.ascii_uppercase
    while k > 26: k -= 26
    so = ""
    for i in s:
        if i.isalpha() == False: so += i
        elif i.islower() == True:
            nc = chr_l.find(i)+k
            so += chr_l[nc] if nc < 26 else chr_l[nc - 26]
        else:
            nc = chr_u.find(i)+k
```

```
            so += chr_u[nc] if nc < 26 else chr_u[nc - 26]
    return so

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    n = int(input().strip())

    s = input()

    k = int(input().strip())

    result = caesarCipher(s, k)

    fptr.write(result + '\n')

    fptr.close()
```

Ceaser cipher

```
def solveMeFirst(a,b):
    # Hint: Type return a+b below
    return a+b


num1 = int(input())
num2 = int(input())
res = solveMeFirst(num1,num2)
print(res)
```

Solve me first

```
def simpleArraySum(ar):
    # Write your code here
     return sum(ar)

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    ar_count = int(input().strip())

    ar = list(map(int, input().rstrip().split()))

    result = simpleArraySum(ar)

    fptr.write(str(result) + '\n')

    fptr.close()
```

Sample aray sum

```
def compareTriplets(a, b):
    # Write your code here
    alice, bob = 0, 0
    for i in range(3):
        if a[i] < b[i]:
            bob += 1
        elif a[i] > b[i]:
            alice += 1
    return [alice, bob]

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    a = list(map(int, input().rstrip().split()))

    b = list(map(int, input().rstrip().split()))

    result = compareTriplets(a, b)

    fptr.write(' '.join(map(str, result)))
    fptr.write('\n')

    fptr.close()
```

Compare the triplets

```
def aVeryBigSum(ar):
    # Write your code here
    return sum(ar)

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    ar_count = int(input().strip())

    ar = list(map(int, input().rstrip().split()))

    result = aVeryBigSum(ar)

    fptr.write(str(result) + '\n')

    fptr.close()
```

A very big sum

```
def diagonalDifference(arr):
    # Write your code here
    left_to_right = []
    right_to_left = []

    for i in range(n):
        l_2_r = arr[i][i]
        left_to_right.append(l_2_r)
        r_2_l = arr[i][n-i-1]
        right_to_left.append(r_2_l)

    l_2_r_sum = sum(left_to_right)
    r_2_l_sum = sum(right_to_left)

    abs_diff = abs(l_2_r_sum - r_2_l_sum)
    return abs_diff

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    n = int(input().strip())

    arr = []

    for _ in range(n):
        arr.append(list(map(int, input().rstrip().split())))

    result = diagonalDifference(arr)

    fptr.write(str(result) + '\n')

    fptr.close()
```

Diagonal difference

```
def plusMinus(arr):
    # Write your code here
    pos = 0
    neg=0
    zeros=0

    for value in arr:
        if value>0:
            pos+=1
```

```
        elif value<0:
            neg+=1
        else:
            zeros+=1

    arr_len = len(arr)
    print(round(pos/arr_len,6))
    print(round(neg/arr_len,6))
    print(round(zeros/arr_len,6))

if __name__ == '__main__':
    n = int(input().strip())

    arr = list(map(int, input().rstrip().split()))

    plusMinus(arr)
```

Plus minus

```
def staircase(n):
    # Write your code here
    for i in range(1, n+1):
        k = n - i
        print(' '*k + '#'*i)

if __name__ == '__main__':
    n = int(input().strip())

    staircase(n)
```

Staircase

```
def miniMaxSum(arr):
    # Write your code here
    sorted(arr)
    arrSum = sum(arr)
    minSum = arrSum - max(arr)
    maxSum = arrSum - min(arr)
    print(minSum, maxSum)

if __name__ == '__main__':

    arr = list(map(int, input().rstrip().split()))

    miniMaxSum(arr)
```

Mini Max sum

```
def birthdayCakeCandles(candles):
    # Write your code here
    return candles.count(max(candles))

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    candles_count = int(input().strip())

    candles = list(map(int, input().rstrip().split()))

    result = birthdayCakeCandles(candles)

    fptr.write(str(result) + '\n')

    fptr.close()
```

Birthday cake candles

```
def timeConversion(s):
    # Write your code here
    if s[-2:] == 'AM':
        if s[0:2] == '12':
            return '00'+s[2:8]
        else: return s[:8]

    else:
        if s[:2] == '12':
            return s[:8]
        else:
            num = int(s[:2]) + 12
            return str(num)+s[2:8]

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    s = input()

    result = timeConversion(s)

    fptr.write(result + '\n')
```

```
    fptr.close()
```

Time conversion

```
def bigSorting(unsorted):
    # Write your code here
    unsorted.sort(key = lambda x: (len(x), x))
    return unsorted

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    n = int(input().strip())

    unsorted = []

    for _ in range(n):
        unsorted_item = input()
        unsorted.append(unsorted_item)

    result = bigSorting(unsorted)

    fptr.write('\n'.join(result))
    fptr.write('\n')

    fptr.close()
```

Big sorting

```
def introTutorial(V, arr):
    # Write your code here
    for i, num in enumerate(arr):
        if num == V:
            return i

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    V = int(input().strip())

    n = int(input().strip())

    arr = list(map(int, input().rstrip().split()))

    result = introTutorial(V, arr)
```

```
    fptr.write(str(result) + '\n')

    fptr.close()
```

Intro to tutorial challenges

```
def insertionSort1(n, arr):
    # Write your code here
    j = n-1
    store = arr[j]

    for i in range(j, -1, -1):
        if store < arr[i-1] and i >= 1:
            arr[i] = arr[i-1]
            print(' '.join(str(x) for x in arr))
        else:
            arr[i] = store
            print(' '.join(str(x) for x in arr))
            break


if __name__ == '__main__':
    n = int(input().strip())

    arr = list(map(int, input().rstrip().split()))

    insertionSort1(n, arr)
```

Insertion sort

```
def insertionSort2(n, arr):
    # Write your code here
    for i in range(1, n):
        ck = None
        for j in range(i):
            if arr[i]<arr[j]:
                ck = j
                break
        else:
            print(' '.join(map(str, arr)))
        if ck is not None:
            val = arr.pop(i)
```

```
            arr.insert(ck, val)
            print(' '.join(map(str, arr)))

if __name__ == '__main__':
    n = int(input().strip())

    arr = list(map(int, input().rstrip().split()))

    insertionSort2(n, arr)
```

Insertion sort part 2

```
n = int(input())
lst = list(map(int,input().split(' ')))
lst.sort()
for i in lst:
    print(i,end=' ')
```

Correctness and loop invariant

```
def runningTime(arr):
    # Write your code here
    cnt = 0
    for i in range(1, len(arr)):
        while i > 0 and arr[i] < arr[i - 1]:
            cnt += 1
            arr[i - 1], arr[i] = arr[i], arr[i - 1]
            i -= 1
    return cnt

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    n = int(input().strip())

    arr = list(map(int, input().rstrip().split()))

    result = runningTime(arr)

    fptr.write(str(result) + '\n')

    fptr.close()
```

Running time of algorithms

```
def quickSort(arr):
    # Write your code here
    p, *a = arr
    left, center, right = [], [p], []
    for c in a:
        if c<p:
            left.append(c)
        elif c>p:
            right.append(c)
        else:
            center.append(c)

    return left+center+right

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    n = int(input().strip())

    arr = list(map(int, input().rstrip().split()))

    result = quickSort(arr)

    fptr.write(' '.join(map(str, result)))
    fptr.write('\n')

    fptr.close()
```

Quicksort 1

```
def countingSort(arr):
    # Write your code here
    result = [0 for _ in range(100)]
    for item in arr:
        result[item] += 1
    return result

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    n = int(input().strip())

    arr = list(map(int, input().rstrip().split()))

    result = countingSort(arr)

    fptr.write(' '.join(map(str, result)))
```

```
    fptr.write('\n')

    fptr.close()
```

Counting sort 1

```
def countingSort(arr):
    # Write your code here
    arrCount = [0]*100
    for x in arr:
        arrCount[x] += 1
    returnArr = []
    for x in range(100):
        for y in range(arrCount[x]):
            returnArr.append(x)
    return returnArr

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    n = int(input().strip())

    arr = list(map(int, input().rstrip().split()))

    result = countingSort(arr)

    fptr.write(' '.join(map(str, result)))
    fptr.write('\n')

    fptr.close()
```

Counting sort 2

```
def closestNumbers(arr):
    # Write your code here
    ls = sorted(arr)
    min = ls[1] - ls[0]

    for i in range(len(ls) - 1):
        if ls[i + 1] - ls[i] < min:
            min = ls[i + 1] - ls[i]

    result = [(ls[i], ls[i + 1]) for i in range(len(ls) - 1)
```

```
                if ls[i + 1] - ls[i] == min]

    return [item for t in result for item in t]

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    n = int(input().strip())

    arr = list(map(int, input().rstrip().split()))

    result = closestNumbers(arr)

    fptr.write(' '.join(map(str, result)))
    fptr.write('\n')

    fptr.close()
```

Closest numbers