# **Adding Two Numbers**

C++

int addNumbers(float a, float b)

{

float c = a+b;

int d= floor(c);

return d;

}

-------------------------------------------------------------

# **Cutting Metal Surplus**

Java 7/8

static int maxProfit(int costPerCut, int salePrice, List<Integer> lengths) {

int maxLength = 0;

for (int length : lengths) {

if (length > maxLength) {

maxLength = length;

}

}

int maxProfit = 0;

for (int i = 1; i < maxLength; i++) {

int sumOfLengths = 0;

int sumOfCutCounts = 0;

int sumOfCutWastes = 0;

for (int length : lengths) {

sumOfLengths += length;

if (length % i == 0) {

sumOfCutCounts += (length/i - 1);

} else {

sumOfCutCounts += (length/i);

}

sumOfCutWastes += (length%i);

}

int profit = sumOfLengths\*salePrice - sumOfCutCounts\*costPerCut - sumOfCutWastes\*salePrice;

if (profit > maxProfit) {

maxProfit = profit;

}

}

return maxProfit;

}

**-------------------------------------------------------------**

# **Permutations Divisible by 8**

**Python 3**

from itertools import permutations as pr

def solve(n):

p = list(pr(n, 3))

for i in p:

if (int(''.join(i)) % 8 == 0):

return 1

return 0

for \_ in range(int(input())):

n = input()

if len(n) <= 2:

n = list(n)

if len(n) == 1 and int(''.join(n)) % 8 == 0:

print('YES')

elif len(n) == 2 and (int(''.join(n)) % 8 == 0 or int(''.join(reversed(n))) % 8 == 0):

print('YES')

else:

print('NO')

continue

if solve(n):

print('YES')

else:

print('NO')

**-------------------------------------------------------------**

# **Efficient Janitor**

**Python 3**

def efficientJanitor(weight):

# Write your code here

count = 0

i,j = 0,len(weight)-1

weight.sort()

while i<=j:

count+=1

if weight[i] + weight[j] <= 3:

i+=1

j -= 1

return count

**-------------------------------------------------------------**

# **Character Reprogramming**

**C++14**

int getMaxDeletions(string s) {

int x=0,y=0,count=0;

for(char ch :s){

if(ch=='R')

{

x++;

count++;

}

else if(ch=='L'){

x--;

count ++;

}

else if(ch=='U'){

y++;

count++;

}

else if(ch=='D'){

y--;

count++;

}

}

return count-abs(x)-abs(y);

}

}

return count-abs(x)-abs(y);

}

**-------------------------------------------------------------**

# **Conference Schedule**

**Python 3**

def maxPresentations(scheduleStart, scheduleEnd):

# Write your code here

sl = sorted((list(zip(\*[scheduleStart, scheduleEnd]))), key = lambda x:x[1])

sl = list(zip(\*sl))

lim = sl[1][0]

n=1

for i in range(1, len(scheduleEnd)):

if sl[0][i] >=lim:

n=n+1

lim = sl[1][i]

return n

**-------------------------------------------------------------**

# **Are they pangrams**

**Python 2**

from string import lowercase, lower

print ["not pangram", "pangram"][lowercase == ''.join(sorted(list(set(raw\_input().lower())-set(' '))))]

# Python 2

from string import lowercase, lower

s = raw\_input().lower() # lowercase input

s = list(set(s)-set(' ')) # remove ' ' spaces

s = ''.join(sorted(s)) # joined the sorted list of unique charaters

print ["not pangram", "pangram"][lowercase == s]

**-------------------------------------------------------------**

# **Balancing Paranthesis**

**C++**

#include <bits/stdc++.h>

using namespace std;

// Function to return required minimum number

int minParentheses(string p)

{

// maintain balance of string

int bal = 0;

int ans = 0;

for (int i = 0; i < p.length(); ++i) {

bal += p[i] == '(' ? 1 : -1;

// It is guaranteed bal >= -1

if (bal == -1) {

ans += 1;

bal += 1;

}

}

return bal + ans;

}

// Driver code

int main()

{

string p;

cin>>p;

// Function to print required answer

cout << minParentheses(p);

return 0;

}

**-------------------------------------------------------------**

# **Dam Design**

**Python 3**

def maxHeight(wallPositions, wallHeights):

# Write your code here

n = len(wallPositions)

mud\_max = 0

for i in range(0, n - 1):

if wallPositions[i] < (wallPositions[i + 1] - 1):

# We have a gap

heightDiff = abs(wallHeights[i + 1] - wallHeights[i])

gapLen = wallPositions[i + 1] - wallPositions[i] - 1

localMax = 0

if gapLen > heightDiff:

low = max(wallHeights[i + 1], wallHeights[i]) + 1

remainingGap = gapLen - heightDiff - 1

localMax = low + remainingGap / 2

else:

localMax = min(wallHeights[i + 1], wallHeights[i]) + gapLen

mud\_max = max(mud\_max, localMax)

return int(mud\_max)

**-------------------------------------------------------------**

# **Duplicated Products**

public static int numDuplicates(List<String> name, List<Integer> price, List<Integer> weight) {

Set<String> uniqueProducts = new HashSet<String>();

for (int i = 0; i < name.size(); i++)

uniqueProducts.add(name.get(i) + " " + price.get(i) + " " + weight.get(i));

return name.size() - uniqueProducts.size();

}

**-------------------------------------------------------------**

# **4th Bit**

**Python 3**

def fourthBit(num):

number = bin(num).replace("0b", "")

string1 = str(number)

return string1[-4]

**-------------------------------------------------------------**

# **Balanced Array**

**Python 3**

def balancedSum(arr):

n=len(arr)

prefixSum = [0] \* n

prefixSum[0] = arr[0]

for i in range(1, n) :

prefixSum[i] = prefixSum[i - 1] + arr[i]

suffixSum = [0] \* n

suffixSum[n - 1] = arr[n - 1]

for i in range(n - 2, -1, -1) :

suffixSum[i] = suffixSum[i + 1] + arr[i]

for i in range(1, n - 1, 1) :

if prefixSum[i] == suffixSum[i] :

return i

return -1

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **Triangle or Not -**

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

static boolean solve(int a,int b,int c){

int[] arr=new int[]{a,b,c};

Arrays.sort(arr);

if(arr[0]+arr[1]>arr[2])

return true;

return false;

}

static String[] triangleOrNot(int[] a, int[] b, int[] c) {

int n=a.length;

String[] res=new String[n];

for(int i=0;i<n;i++){

res[i]=solve(a[i],b[i],c[i])==true?"Yes":"No";

}

return res;

}

public static void main(String[] args) throws IOException {

Scanner in = new Scanner(System.in);

final String fileName = System.getenv("OUTPUT\_PATH");

BufferedWriter bw = null;

if (fileName != null) {

bw = new BufferedWriter(new FileWriter(fileName));

}

else {

bw = new BufferedWriter(new OutputStreamWriter(System.out));

}

String[] res;

int a\_size = 0;

a\_size = Integer.parseInt(in.nextLine().trim());

int[] a = new int[a\_size];

for(int i = 0; i < a\_size; i++) {

int a\_item;

a\_item = Integer.parseInt(in.nextLine().trim());

a[i] = a\_item;

}

int b\_size = 0;

b\_size = Integer.parseInt(in.nextLine().trim());

int[] b = new int[b\_size];

for(int i = 0; i < b\_size; i++) {

int b\_item;

b\_item = Integer.parseInt(in.nextLine().trim());

b[i] = b\_item;

}

int c\_size = 0;

c\_size = Integer.parseInt(in.nextLine().trim());

int[] c = new int[c\_size];

for(int i = 0; i < c\_size; i++) {

int c\_item;

c\_item = Integer.parseInt(in.nextLine().trim());

c[i] = c\_item;

}

res = triangleOrNot(a, b, c);

for(int res\_i = 0; res\_i < res.length; res\_i++) {

bw.write(String.valueOf(res[res\_i]));

bw.newLine();

}

bw.close();

}

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **Duplicate products**

int numDuplicates(vector<string> name, vector<int> price, vector<int> weight) {

int count=0;

string product="";

unordered\_map<string,int>freq;

for(int i=0;i<name.size();i++)

{

product=name[i]+" "+to\_string(price[i])+" "+to\_string(weight[i]);

if(freq[product])

{

count++;

}

else

{

freq[product]++;

}

}

return count;

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **Circular Printer**

sum=0

sum+=(min(abs(ord('A')-ord(s[0])),26-abs(ord('A')-ord(s[0]))))

for i in range(len(s)-1):

sum+=(min(abs(ord(s[i])-ord(s[i+1])),26-abs(ord(s[i])-ord(s[i+1]))))

return sum

Bit Logic

flag = 0

while(lo<hi):

for i in range(lo+1,hi+1):

temp = lo^i

if(temp>flag and temp<=k):

flag = temp

lo+=1

return flag

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **Largest String**

#!/bin/python3

import math

import os

import random

import re

import sys

#

# Complete the 'getLargestString' function below.

#

# The function is expected to return a STRING.

# The function accepts following parameters:

# 1. STRING s

# 2. INTEGER k

#

def getLargestString(word, k):

countArr = [0]\*26

a, ans = ord('a'), []

for c in word:

countArr[ord(c)-a] += 1

i = 25 # start at z

# Now we have count of all chars we start from z to a.

while i >= 0:

# More chars than the window permits

if countArr[i] > k:

# Lets append k letters if they exist.

letter = chr(i+a)

ans.append(letter\*k)

countArr[i] -= k

# look for the next element

j = i-1

while(countArr[j] <= 0 and j>0):

j -= 1

# add one of the next element

if countArr[j] > 0 and j >= 0:

letter = chr(j+a)

ans.append(letter)

countArr[j] -= 1

else:

break # we cant build string more.

elif countArr[i] > 0:

letter = chr(i+a)

ans.append(letter\*countArr[i])

countArr[i] = 0

else: # this letter we can't do anything, lets skip

i -= 1

# print(''.join(ans))

return ''.join(ans)

# Write your code here

if \_name\_ == '\_main\_':

fptr = open(os.environ['OUTPUT\_PATH'], 'w')

s = input()

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

result = getLargestString(s, k)

fptr.write(result + '\n')

fptr.close()

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **Character Reprogramming**

**C**

int getMaxDeletions(char\* s)

{

int i = 0, size = 0;

while(s[i] != '\0')

{

size++;

i++;

}

i = 0;

int \*array;

array = calloc(size, sizeof(int));

while(s[i] != '\0')

{

if(s[i] == 'U')

{

array[i] = 1;

}

else if(s[i] == 'D')

{

array[i] = -1;

}

else if(s[i] == 'R')

{

array[i] = 2;

}

else if(s[i] == 'L')

{

array[i] = -2;

}

i++;

}

int Ucounter = 0; int Rcounter = 0; int Dcounter = 0; int Lcounter = 0;

i = 0;

while(i < size)

{

if(array[i] == 1)

{

Ucounter++;

}

else if(array[i] == -1)

{

Dcounter++;

}

else if(array[i] == 2)

{

Rcounter++;

}

else if(array[i] == -2)

{

Lcounter++;

}

i++;

}

int answer = 0;

if(Ucounter>= Dcounter)

{

answer = answer + Dcounter;

}

else

{

answer = answer + Ucounter;

}

if(Rcounter >= Lcounter)

{

answer = answer+Lcounter;

}

else

{

answer = answer+Rcounter;

}

return 2\*answer;

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **Reverse Queries**

for i in operations:

x = i[0]

y = i[1]

temp = arr[x:y+1]

temp = temp[::-1]

for i in range(x,y+1):

arr[i] = temp[i-x]

return arr

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **Cutting Metal Surpulus**

#include <bits/stdc++.h>

using namespace std;

int solve(vector<int> &rods, int sz, int cpc, int sl) {

int pr = 0;

for(int r : rods) {

int temp = 0;

if(r%sz == 0) {

temp += ((r/sz) \* sz \* sl) - (r/sz - 1) \* cpc;

} else {

temp += ((r/sz) \* sz \* sl) - (r/sz) \* cpc;

}

if(temp > 0) pr += temp;

}

return pr;

}

int main() {

int n;

int cpc , sl;

cin >> cpc >> sl;

cin >> n;

vector<int> v(n);

int maxlen = 0;

for(int i=0; i<n; i++) {

cin >> v[i];

maxlen = max(maxlen, v[i]);

}

int ans = INT\_MIN;

for(int sz=1; sz<=maxlen; sz++) {

int prof = solve(v, sz, cpc, sl);

ans = max(prof, ans);

}

cout <<ans << endl;

return 0;

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **Fixbuzz**

def fizzbuzz(n):

if n % 3 == 0 and n % 5 == 0:

print('FizzBuzz')

elif n % 3 == 0:

print('Fizz')

elif n % 5 == 0:

print('Buzz')

else:

print(n)

x=(int)(input())

for i in range(1,x+1):

fizzbuzz(i)

# 

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **find the factor**

from math import sqrt

l=[]

def pthFactor(n, p):

k=int(sqrt(n))+1

for i in range(1,k,1):

if n%i==0:

l.append(i)

if(i!=sqrt(n)):

l.append(int(n/i))

l.sort(reverse=False)

if(p>len(l)):

print("0")

else:

print(l[p-1])

n=int(input())

p=int(input())

pthFactor(n,p)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **Condensed List**

SinglyLinkedListNode\* condense(SinglyLinkedListNode\* head) {

struct SinglyLinkedListNode \*p;

unordered\_set<int> s;

p=head;

s.insert(head->data);

while(p!=NULL && p->next!=NULL)

{

if(s.find(p->next->data)==s.end())

{

s.insert(p->next->data);

p=p->next;

}

else {

p->next=p->next->next;

}

}

for (auto it = s.begin(); it !=s.end(); ++it)

cout << ' ' << \*it;

return head;

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# 

# **No paired allowed**

def minimalOperations(word\_collection):

counter = []

for words in word\_collection:

words = list(words)

count = 0

i = 0

while i < len(words)-1:

if words[i] == words[i+1]:

count += 1

i += 1

i += 1

counter.append(count)

return counter

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **Product Sort**

def itemsSort(items):

l=items.copy()

r=[]

s=[]

l=set(l)

for i in l:

c=items.count(i)

s.append([c,i])

s.sort(key=lambda x:x[0])

for i in s:

q=i[0]

while q!=0:

r.append(i[1])

q-=1

return r

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **arrange the words**

import re

sentence\_rgx = re.compile(r'^[A-Z][a-z ]\*\.$')

# satisfy constraints

\_sentence = str(sentence)

assert len(\_sentence) >= 1 and len(\_sentence) <= 10\*\*5

assert re.match(sentence\_rgx, \_sentence)

# split the sentence into a list of words, then

# decapitalize each word, remove full stop and

# strip excess whitespace in between words

words = [

word.lower()[0:len(word)-1] if word[-1] == '.' else word.lower()

for word in re.sub(r'[ ]+', ' ', \_sentence).split(' ')

]

# sort by length of words - note that by default Python

# implements Timsort, and therefore is stable (ie. order

# of pre-sorted words are retained)

words.sort(key=len)

# capitalize first word and add full stop to last word

words[0] = f'{words[0][0].upper()}{words[0][1:]}'

words[-1] = f'{words[-1]}.'

# join words into a sentence, then do

# one last sanity check

arranged = ' '.join(words)

assert re.match(sentence\_rgx, arranged)

return arranged

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **minimum difference sum**

#!/bin/python3

import math

import os

import random

import re

import sys

#

# Complete the 'minDiff' function below.

#

# The function is expected to return an INTEGER.

# The function accepts INTEGER\_ARRAY arr as parameter.

#

def minDiff(arr):

# Write your code here

sum\_=0

arr.sort()

for i in range(len(arr)-1):

sum\_+=arr[i+1]-arr[i]

return sum\_

if \_name\_ == '\_main\_':

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **Maximum index**

int maxIndex(int steps, int badIndex) {

int i=0;

int j=1;

int tempStep = steps;

int scene1, scene2;

while (steps--) {

if (i+j != badIndex)

i = i+j;

j++;

}

scene1 = i;

i = 0;

tempStep = tempStep - 1;

j = 2;

while (tempStep--) {

if (i+j != badIndex)

i = i+j;

j++;

}

scene2 = i;

return scene1 > scene2 ? scene1 : scene2;

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **Product Defects**

def largestArea(samples):

# `T[i][j]` stores the size of maximum square submatrix ending at `M[i][j]`

T = [[0 for x in range(len(samples[0]))] for y in range(len(samples))]

# `max` stores the size of the largest square submatrix of 1's

max = 0

# fill in a bottom-up manner

for i in range(len(samples)):

for j in range(len(samples[0])):

T[i][j] = samples[i][j]

# if we are not at the first row or first column and the

# current cell has value 1

if i > 0 and j > 0 and samples[i][j] == 1:

# the largest square submatrix ending at `M[i][j]` will be 1 plus

# minimum of the largest square submatrix ending at `M[i][j-1]`,

# `M[i-1][j]` and `M[i-1][j-1]`

T[i][j] = min(T[i][j - 1], T[i - 1][j], T[i - 1][j - 1]) + 1

# update maximum size found so far

if max < T[i][j]:

max = T[i][j]

# return size of the largest square matrix

return max

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **Maximizing the final element**

int getMaxValue(vector<int> arr) {

int n= arr.size();

sort(arr.begin() , arr.end());

// If the first element

// is not equal to 1

if (arr[0] != 1)

arr[0] = 1;

// Traverse the array to make

// difference between adjacent

// elements <=1

for (int i = 1; i < n; i++) {

if (arr[i] - arr[i - 1] > 1) {

arr[i] = arr[i - 1] + 1;

}

}

return arr[n - 1];