P1007 还是最大公约数

#include <stdio.h>

// 函数声明：计算两个数的最大公约数

int gcd(int a, int b) {

if (b == 0) {

return a;

} else {

return gcd(b, a % b);

}

}

int main() {

int n;

scanf("%d", &n); // 输入数据组数

while (n--) {

int a, b, c;

scanf("%d %d %d", &a, &b, &c); // 输入三个数

// 计算 a 和 b 的最大公约数

int ab\_gcd = gcd(a, b);

// 计算 ab\_gcd 和 c 的最大公约数，即为 a,b,c 的最大公约数

int result = gcd(ab\_gcd, c);

printf("%d\n", result); // 输出结果

}

return 0;

}

# P1027 坏人必须死

#### 题目描述

有 *m* 个好人和 *m* 个坏人坐成一个圈，前 *m* 个人是好人（编号为 1，2，到*m*） , 后 *m* 个人是坏人（编号为 *m*+1,*m*+2到,2*m*）. 现在他们开始循环报数，要求从编号为 1 的人开始报，如果有人报到 *k* 则他必须死亡，而死掉的人的下一个人则继续从 1开始报数。请求出最小的 *k* 使得，所有的坏人都死掉而好人都活下来.

#### 输入描述

多组输入，每组输入 1 行，为一个整数 *m* (1≤*m*≤12)

#### 输出描述

对于每组输入，输出为 1 个整数 *k* 为使得坏人都死掉的最小报的数

#include<stdio.h>

int ysf(int sum, int k, int n){

if(n == 1){

return (sum + k - 1) % sum + 1;

}

return (ysf(sum-1,k,n-1) + k - 1) % sum + 1;

}

int main(){

int m;

while(scanf("%d", &m) != EOF){

int sum = 2 \* m, n = 0;

int k = m + 1;

while(++n <= m){

int t = ysf(sum,k,n);

if(t <= m){

n = 0;

k++;

}

}

printf("%d\n", k);

}

return 0;

}

# P1073 数字反转\*

#### 题目描述

给定一个整数，请将该数各个位上数字反转得到一个新数。新数也应满足整数的常见形式，即除非给定的原数为零，否则反转后得到的新数的最高位数字不应为零。数据范:  −1050≤*N*≤1050,*N*!=0

#### 输入描述

每 1行一个整数 *N*

#### 输出描述

每 1 行一个整数，表示反转后的新数.

#include <stdio.h>

#include <string.h>

// 函数用于反转字符串

void reverseString(char \*str) {

int left = 0;

int right = strlen(str) - 1;

while (left < right) {

char temp = str[left];

str[left] = str[right];

str[right] = temp;

left++;

right--;

}

}

int main() {

char str[100]; // 假设输入的整数字符串长度不超过 100

while (scanf("%s", str) != EOF) {

// 排除特殊情况，即输入为 "0" 时

if (strcmp(str, "0") == 0) {

printf("0\n");

} else {

// 处理负数的情况

int isNegative = 0;

if (str[0] == '-') {

isNegative = 1;

memmove(str, str + 1, strlen(str)); // 移除负号

}

reverseString(str); // 反转字符串

// 去除前导零

int i = 0;

while (str[i] == '0') {

i++;

}

// 输出结果

printf("%s%s\n", (isNegative ? "-" : ""), str + i);

}

}

return 0;

}

#### 题目描述

Give you a n∗*n* matrix, if you rotate it 90 degrees(clockwise), we call it 1-Matrix, if you rotate it 180 degrees, we call it 2-Matrix, etc … This is said if you rotate the Matrix 90∗*k* degrees, then we call it k-Matrix. Now, your task is to calculate all the sum of i-Matrix (0<= i<= k).

#### 输入描述

There multiple test cases. Each test case begins with one integer n(1 <= n <= 10), following n lines, each line contains n integers, describe the original matrix, then a single line contains a k (1 <= k <= 10^8)described above. Process to end of file.

#### 输出描述

For each case, output the sum of all the i-Matrix (0<= i<= k) in n lines. Each line there are n integers separated by one space. Note that there is no extra space at the end of each line.

Attention：for the largest data，you‘d better use the type of long long to ouput your answer。

#### 样例输入

3

1 2 3

2 3 4

3 4 5

10

#### 样例输出

33 32 31

34 33 32

35 34 33

#include<stdio.h>

#define N 10

void rotation(long long int a[][10],int d);

int main()

{

int n;

while((scanf("%d",&n))!=EOF)

{

int k;

long long int x[N][N]={0};

long long int y[N][N]={0};

long long int z[N][N]={0}; //这里要对x和n初始化不然会输出奇怪的数字

int i=0,j=0;

int m=0;

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

{

for(j=0;j<n;j++)

{

scanf("%lld",&x[i][j]);

}

}

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

{

for(j=0;j<n;j++)

{

z[i][j]=x[i][j];

}

}

scanf("%d",&k);

for(m=1;m<=4;m++)

{

rotation(x,n);

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

for(j=0;j<n;j++)

y[i][j]+=x[i][j];

}

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

for(j=0;j<n;j++)

y[i][j]=(k/4)\*y[i][j];

for(m=1;m<=(k%4);m++)

{

rotation(x,n);

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

for(j=0;j<n;j++)

y[i][j]+=x[i][j];

}

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

{

for(j=0;j<n;j++)

{

printf("%lld ",y[i][j]+z[i][j]);

}

printf("\n");

}

}

return 0;

}

void rotation(long long int a[][10],int d)

{

int p,q;

long long int b[N][N]={0};

for(p=0;p<d;p++)

for(q=0;q<d;q++)

{

b[p][q]=a[d-1-q][p];

}

for(p=0;p<d;p++)

for(q=0;q<d;q++)

{

a[p][q]=b[p][q];

}

}

#### 题目描述

任意一个每位数字不完全相同的 4 位正整数，将各位上数字重新组合成一个最大数和一个最小数相减，重复这个过程，必定得到 6174

#### 输入描述

有多组测试数据，每组测试数据占一行.

每行输入一个 4 位正整数.

#### 输出描述

占一行，最终得到 6174 的变换次数最少的

#include<stdio.h>

#include<string.h>

int main()

{

int i,j,a,b,c,n[4],s;

while (scanf("%d", &c) != EOF) {

int cnt = 0;

do

{

if(c==6174)

{

printf("%d\n", cnt);

return 0;

}

n[0]=c/1000;

n[1]=c/100%10;

n[2]=c/10%10;

n[3]=c%10;

for(i=0;i<3;i++)

for(j=0;j<3;j++)

if(n[j]<n[j+1])

{

s=n[j];

n[j]=n[j+1];

n[j+1]=s;

}

a=n[0]\*1000+n[1]\*100+n[2]\*10+n[3];

b=n[3]\*1000+n[2]\*100+n[1]\*10+n[0];

c=a-b;

cnt++;

}while(c!=6174);

printf("%d\n", cnt);

}

}

# P1108 二进制转化为十六进制

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// 函数用于将四位二进制数转换为十六进制字符

char binaryToHex(char \*binary) {

// 将二进制字符串转换为整数

int decimal = strtol(binary, NULL, 2);

// 根据整数值返回对应的十六进制字符

if (decimal >= 0 && decimal <= 9) {

return decimal + '0';

} else {

return decimal - 10 + 'A';

}

}

int main() {

char binary[33];

while (1) {

// 读取输入的二进制数

if (scanf("%s", binary) == EOF || strcmp(binary, "0") == 0) {

break;

}

// 补齐输入的二进制数为32位，确保不以0开头

int len = strlen(binary);

int padding = 32 - len;

if (padding > 0) {

memmove(binary + padding, binary, len + 1); // 包括'\0'字符

memset(binary, '0', padding);

}

// 将二进制数每四位一组转换为对应的十六进制字符

char hex[9];

for (int i = 0; i < 32; i += 4) {

char binaryGroup[5];

strncpy(binaryGroup, binary + i, 4);

binaryGroup[4] = '\0';

hex[i / 4] = binaryToHex(binaryGroup);

}

hex[8] = '\0';

// 输出转换后的十六进制数，去掉开头的0

int start = 0;

while (hex[start] == '0') {

start++;

}

if (start == 8) {

printf("0\n"); // 全部为0时输出0

} else {

printf("%s\n", hex + start);

}

}

return 0;

}

[P1117](https://client.vpn.nuist.edu.cn/https/webvpn893ff9021738b0357186c0f23fc2aed6e24ca283e886022bc5d861ea12f03963/course/94/submission) 运动会开幕式入场

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_COUNTRY\_LENGTH 100 // 国家名的最大长度

// 比较函数，用于排序

int compare(const void \*a, const void \*b) {

return strcmp(\*(const char \*\*)a, \*(const char \*\*)b);

}

int main() {

int n;

char \*\*countries; // 存储国家名的指针数组

// 输入国家数

scanf("%d", &n);

// 分配内存

countries = (char \*\*)malloc(n \* sizeof(char \*));

if (countries == NULL) {

fprintf(stderr, "Memory allocation failed\n");

return 1;

}

// 输入国家名

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

countries[i] = (char \*)malloc(MAX\_COUNTRY\_LENGTH \* sizeof(char));

if (countries[i] == NULL) {

fprintf(stderr, "Memory allocation failed\n");

return 1;

}

scanf("%s", countries[i]);

}

// 将中国移到数组末尾

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

if (strcmp(countries[i], "China") == 0) {

char \*temp = countries[i];

countries[i] = countries[n-1];

countries[n-1] = temp;

break;

}

}

// 对国家名进行排序

qsort(countries, n-1, sizeof(countries[0]), compare);

// 输出排序后的国家名

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

printf("%s\n", countries[i]);

}

// 释放内存

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

free(countries[i]);

}

free(countries);

return 0;

}

# P1118 ISBN号

#include <stdio.h>

#include <ctype.h>

// 计算识别码

int computeChecksum(char \*isbn) {

int sum = 0;

int weight = 1;

for (int i = 0; i < 9; i++) {

if (isdigit(isbn[i])) {

sum += (isbn[i] - '0') \* weight;

weight++;

}

}

return sum % 11;

}

int main() {

int T;

scanf("%d", &T);

while (T--) {

char isbn[14];

scanf("%s", isbn);

// 计算输入的ISBN号码的识别码

int givenChecksum = (isbn[12] == 'X') ? 10 : (isbn[12] - '0');

// 计算正确的识别码

int correctChecksum = computeChecksum(isbn);

// 如果识别码正确，则输出 "Right"，否则输出正确的ISBN号码

if (givenChecksum == correctChecksum) {

printf("Right\n");

} else {

// 将分隔符位置的识别码排除在计算之外

int weight = 1;

int sum = 0;

for (int i = 0; i < 13; i++) {

if (isdigit(isbn[i]) && isbn[i] != '-') {

sum += (isbn[i] - '0') \* weight;

weight++;

}

}

correctChecksum = sum % 11;

// 根据正确的识别码更新输入的识别码

if (correctChecksum == 10) {

isbn[12] = 'X';

} else {

isbn[12] = correctChecksum + '0';

}

printf("%s\n", isbn);

}

}

return 0;

}

# P1120 大整数加法

#include <stdio.h>

#include <string.h>

#define MAX\_LENGTH 201

// 将字符串表示的数字转换为整数数组

void convertToDigits(char \*numStr, int \*digits) {

int len = strlen(numStr);

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

digits[i] = numStr[len - 1 - i] - '0';

}

}

// 大整数加法

void addBigIntegers(int \*num1, int \*num2, int len1, int len2) {

int result[MAX\_LENGTH] = {0};

int carry = 0;

int maxLen = (len1 > len2) ? len1 : len2;

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

int sum = num1[i] + num2[i] + carry;

result[i] = sum % 10;

carry = sum / 10;

}

if (carry) {

result[maxLen] = carry;

maxLen++;

}

for (int i = maxLen - 1; i >= 0; i--) {

printf("%d", result[i]);

}

printf("\n");

}

int main() {

int n;

scanf("%d", &n);

while (n--) {

char num1[MAX\_LENGTH], num2[MAX\_LENGTH];

scanf("%s %s", num1, num2);

int digits1[MAX\_LENGTH] = {0}, digits2[MAX\_LENGTH] = {0};

convertToDigits(num1, digits1);

convertToDigits(num2, digits2);

int len1 = strlen(num1);

int len2 = strlen(num2);

addBigIntegers(digits1, digits2, len1, len2);

}

return 0;

}

# P1121 大整数减法

#include <stdio.h>

#include <string.h>

#define MAX\_LENGTH 201

// 将字符串表示的数字转换为整数数组

void convertToDigits(char \*numStr, int \*digits) {

int len = strlen(numStr);

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

digits[i] = numStr[len - 1 - i] - '0';

}

}

// 大整数减法

void subtractBigIntegers(int \*num1, int \*num2, int len1, int len2) {

int result[MAX\_LENGTH] = {0};

int borrow = 0;

for (int i = 0; i < len1; i++) {

int diff = num1[i] - (i < len2 ? num2[i] : 0) - borrow;

if (diff < 0) {

diff += 10;

borrow = 1;

} else {

borrow = 0;

}

result[i] = diff;

}

// 找到结果的最高有效位

int maxLen = len1;

while (maxLen > 0 && result[maxLen - 1] == 0) {

maxLen--;

}

// 输出结果

for (int i = maxLen - 1; i >= 0; i--) {

printf("%d", result[i]);

}

printf("\n");

}

int main() {

char num1[MAX\_LENGTH], num2[MAX\_LENGTH];

while (scanf("%s", num1) != EOF) {

scanf("%s", num2);

int digits1[MAX\_LENGTH] = {0}, digits2[MAX\_LENGTH] = {0};

convertToDigits(num1, digits1);

convertToDigits(num2, digits2);

int len1 = strlen(num1);

int len2 = strlen(num2);

// 大整数减法

subtractBigIntegers(digits1, digits2, len1, len2);

}

return 0;

}

# P1128 约瑟夫问题

#### 题目描述

已知 n 个人（编号分别为 1、2、3，……、n）围坐在一张圆桌周围，从编号为 1 的人开始报数，数到 m 的那个人出列；他的下一个人又从 1 开始报数，数到 m 的那个人又出列，依次规律重复下去，直到圆桌周围的人全部出列

#### 输入描述

一行：人数 n 和间隔数 m，均不大于 100。

#### 输出描述

出列顺序，每个编号之间用一个空格分开

#include <stdio.h>

// 模拟约瑟夫环问题

void josephus(int n, int m) {

int i, j, count;

int a[101];

// 初始化编号

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

a[i] = i + 1;

}

// 开始报数并出列

count = 0;

j = 0;

while (count < n) {

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

if (a[i] > 0) {

j++;

if (j == m) {

printf("%d ", a[i]);

a[i] = 0;

count++;

j = 0;

}

}

}

}

printf("\n");

}

int main() {

int n, m;

while (scanf("%d %d", &n, &m) != EOF) {

josephus(n, m);

}

return 0;

}

# P1129 走出泥潭

#### 题目描述

探险队要穿越泥潭，必须选择可踩踏的落脚点。可是泥潭面积很大，落脚点又实在少得可怜，一不小心就会深陷泥潭而无法脱身。侦查员费尽周折才从老乡手里弄到了一份地图，图中标出了落脚点的位置，而且令人震惊的是：泥潭只有一条穿越路线，且对于 n××m 的地图，其中 n 和 m 均小于等于 50。路线长度为 n+m-1！请编程为探险队找出穿越路线。

#### 输入描述

两个整数 n 和 m，表示泥潭的长和宽。下面 n 行 m 列表示地形（0 表示泥潭，1 表示落脚点）

#### 输出描述

用坐标表示穿越路线，坐标之间用 > 分隔

#### 样例输入

6 9

1 1 1 0 0 0 0 0 0

0 0 1 1 1 0 0 0 0

0 0 0 0 1 0 0 0 0

0 0 0 0 1 1 0 0 0

0 0 0 0 0 1 1 1 1

0 0 0 0 0 0 0 0 1

#### 样例输出

(1,1)>(1,2)>(1,3)>(2,3)>(2,4)>(2,5)>(3,5)>(4,5)>(4,6)>(5,6)>(5,7)>(5,8)>(5,9)>(6,9)

#include<stdio.h>

#include<string.h>

#define N 101

int map[N][N];

int n,m;

struct point{

int l,r;

}node[N];

int ans;

void DFS(int x,int y) {

if(x == n && y == m) {

for(int i = 1; i < ans; i++)

printf("(%d,%d)>",node[i].l,node[i].r);

printf("(%d,%d)\n",x,y);

} else {

if(map[x][y] == 1 && x <= n && y <= m) {

node[ans].l = x;

node[ans].r = y;

ans++;

DFS(x + 1,y);

DFS(x,y + 1);

}

}

}

int main() {

while(scanf("%d %d",&n,&m) != EOF) {

ans = 1;

memset(map,0,sizeof(map));

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

for(int j = 1; j <= m; j++)

scanf("%d",&map[i][j]);

DFS(1,1);

}

return 0;

}

# P1131 子串

#include<stdio.h>

#include<string.h>

void reverse(char \*s, int n) {

for(int i = 0, j = n - 1; i < j; i++, j--) {

char c = s[i];

s[i] = s[j];

s[j] = c;

}

}

int main() {

int t;

scanf("%d", &t);

while(t--) {

int n, i, j, k, min = 1000, max = 0, i\_temp, result = 0;

char str[10][100];

scanf("%d", &n);

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

scanf("%s", str[i]);

if(strlen(str[i]) < min) {

min = strlen(str[i]);

i\_temp = i;

}

}

for(i = 0; i < min - 1; i++) {

for(j = min - i; j > 0; j--) {

int flag1 = 1, flag2 = 0;

char temp[100] = {0};

memcpy(temp, str[i\_temp] + i, j);

for(k = 0; k < n; k++) {

if(strstr(str[k], temp) == NULL) {

reverse(temp, strlen(temp));

if(strstr(str[k], temp) == NULL) {

break;

}

}

}

if(strlen(temp) > max && k == n) {

max = strlen(temp);

}

}

}

printf("%d\n", max);

}

return 0;

}

# P1134 高精度阶乘

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_DIGITS 3000

void multiply(char result[], int size, int multiplier) {

int carry = 0;

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

int product = (result[i] - '0') \* multiplier + carry;

result[i] = (product % 10) + '0';

carry = product / 10;

}

while (carry > 0) {

result[size++] = (carry % 10) + '0';

carry /= 10;

}

result[size] = '\0';

}

void factorial(int n) {

char result[MAX\_DIGITS];

memset(result, '0', sizeof(result));

result[0] = '1';

int size = 1;

for (int i = 2; i <= n; i++) {

multiply(result, size, i);

size = strlen(result);

}

for (int i = size - 1; i >= 0; i--) {

printf("%c", result[i]);

}

printf("\n");

}

int main() {

int n;

while (scanf("%d", &n) != EOF) {

factorial(n);

}

return 0;

}

# P1136 高精度乘法

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

int main() {

char a[101], b[101], swap[101];

while (scanf("%s %s", a, b) != EOF) {

int a1[101] = { 0 }, b1[101] = { 0 };

int lena1, lenb1;

lena1 = strlen(a), lenb1 = strlen(b);

int n;

n = (lena1 > lenb1 ? lena1 : lenb1);

int \*p;

p = (int\*)malloc(sizeof(int) \* (2 \* n));

memset(p, 0, sizeof(int) \* (2 \* n));

if (lena1 < lenb1) {

strcpy(swap, b);

strcpy(b, a);

strcpy(a, swap);

}

lena1 = strlen(a), lenb1 = strlen(b);

for (int i1 = lena1 - 1; i1 >= 0; i1--) {

a1[lena1 - 1 - i1] = a[i1] - '0';

}

for (int k1 = lenb1 - 1; k1 >= 0; k1--) {

b1[lenb1 - 1 - k1] = b[k1] - '0';

}

int len;

for (int i = 0; i < lenb1; i++) {

len = i;

for (int k = 0; k < lena1; k++) {

p[len] += b1[i] \* a1[k];

if (p[len] > 9) {

p[len + 1] += p[len] / 10;

p[len] = p[len] % 10;

}

len++;

}

}

while (p[len] == 0) {

len--;

}

for (int k2 = len; k2 >= 0; k2--) {

printf("%d", p[k2]);

}

free(p);

printf("\n");

}

return 0;

}

# P1139 再分麦粒

#include <stdio.h>

int main() {

int n;

while(~scanf("%d", &n))

{

int rsl[500]={0}, temp = 0, cnt = 1, j = 0, i = 0;

rsl[0] = 1;

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

{

temp = 0;

for(j = 0; j < cnt; j++)

{

temp = rsl[j] \* 2 + temp;

rsl[j] = temp % 10;

temp /= 10;

}

if(j<=499){

while(temp)

{

cnt++;

rsl[j++] += temp % 10;

temp /= 10;

}

}

}

for(i = 499; i >= 1; i--)

{

printf("%d", rsl[i]);

}

printf("%d", rsl[0] - 1);

printf("\n");

}

return 0;

}

# P1140 循环数

#include <stdio.h>

#include <stdbool.h>

#include <string.h>

#define MAX\_LENGTH 61

// 函数用于判断一个整数是否是循环数

bool isCircularNumber(char num[]) {

int len = strlen(num);

char product[MAX\_LENGTH]; // 存储乘积的字符串

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

// 计算乘积

int carry = 0;

for (int j = len - 1; j >= 0; j--) {

int digit = (num[j] - '0') \* i + carry;

product[j] = digit % 10 + '0';

carry = digit / 10;

}

if (carry) { // 如果有进位，说明乘积不是N位

return false;

}

// 检查乘积是否是循环字符串

bool found = false;

for (int k = 0; k < len; k++) {

if (strncmp(product, num + k, len - k) == 0 && strncmp(product + len - k, num, k) == 0) {

found = true;

break;

}

}

if (!found) {

return false;

}

}

return true;

}

int main() {

char num[MAX\_LENGTH];

while (scanf("%s", num) != EOF) {

if (isCircularNumber(num)) {

printf("%s is cyclic\n", num);

} else {

printf("%s is not cyclic\n", num);

}

}

return 0;

}

# P1152 时区间时间的转换

#include<stdio.h>

#include<string.h>

double table(char\*s){

if(!strcmp(s,"UTC")) return 0.0;

if(!strcmp(s,"GMT")) return 0.0;

if(!strcmp(s,"BST")) return 1.0;

if(!strcmp(s,"IST")) return 1.0;

if(!strcmp(s,"WET")) return 0.0;

if(!strcmp(s,"WEST")) return 1.0;

if(!strcmp(s,"CET")) return 1.0;

if(!strcmp(s,"CEST")) return 2.0;

if(!strcmp(s,"EET")) return 2.0;

if(!strcmp(s,"EEST")) return 3.0;

if(!strcmp(s,"MSK")) return 3.0;

if(!strcmp(s,"MSD")) return 4.0;

if(!strcmp(s,"AST")) return -4.0;

if(!strcmp(s,"ADT")) return -3.0;

if(!strcmp(s,"NST")) return -3.5;

if(!strcmp(s,"NDT")) return -2.5;

if(!strcmp(s,"EST")) return -5.0;

if(!strcmp(s,"EDT")) return -4.0;

if(!strcmp(s,"CST")) return -6.0;

if(!strcmp(s,"CDT")) return -5.0;

if(!strcmp(s,"MST")) return -7.0;

if(!strcmp(s,"MDT")) return -6.0;

if(!strcmp(s,"PST")) return -8.0;

if(!strcmp(s,"PDT")) return -7.0;

if(!strcmp(s,"HST")) return -10.0;

if(!strcmp(s,"AKST")) return -9.0;

if(!strcmp(s,"AKDT")) return -8.0;

if(!strcmp(s,"AEST")) return 10.0;

if(!strcmp(s,"AEDT")) return 11.0;

if(!strcmp(s,"ACST")) return 9.5;

if(!strcmp(s,"ACDT")) return 10.5;

if(!strcmp(s,"AWST")) return 8.0;

}

int main(){

int n;

scanf("%d",&n);

while(n--){

char s[11],a[5],b[5];

int t,i;

scanf("%s",s);

if(s[0]=='n')

t=12\*60;

else if(s[0]=='m')

t=24\*60;

else{

if(s[1]==':')

t=(s[0]-'0')\*60+(s[2]-'0')\*10+(s[3]-'0');

else

t=((s[0]-'0')\*10+(s[1]-'0'))\*60

+(s[3]-'0')\*10+(s[4]-'0');

if(t/60==12)t-=12\*60;

scanf("%s",s);

if(s[0]=='p')t+=12\*60;

}

scanf("%s%s",a,b);

t+=(int)((table(b)-table(a))\*60);

if(t<=0)t+=24\*60;

if(t>24\*60)t-=24\*60;

if(t==12\*60)

printf("noon\n");

else if(t==24\*60)

printf("midnight\n");

else{

// printf("\*%d %d\n",t/60,t%60);

char ss[6];

strcpy(ss,"a.m.");

if(t>=12\*60){

t-=12\*60;

ss[0]='p';

}

if(t<60)

t+=12\*60;

printf("%d:%d %s\n",t/60,t%60,ss);

}

}

return 0;

}

# P1155 数列特殊求和

#include <stdio.h>

#define MOD 10000

// 快速幂算法求 a^b % MOD

int power(int a, int b) {

int result = 1;

while (b > 0) {

if (b % 2 == 1) {

result = (result \* a) % MOD;

}

a = (a \* a) % MOD;

b /= 2;

}

return result;

}

int main() {

int N;

scanf("%d", &N);

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

int a, b;

scanf("%d %d", &a, &b);

int sum = 0;

for (int j = 1; j <= a; j++) {

sum = (sum + power(j, b)) % MOD;

}

printf("%d\n", sum);

}

return 0;

}

# P1161 Lotto

# P1169 乒乓球

#include<stdio.h>

char a[1000000001];

int main() {

int l = 0;

char c;

scanf("%c", &c);

while (c != 'E') {

a[l] = c;

l++;

scanf("%c", &c);

}

int p1 = 0, p2 = 0;

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

if (a[i] == 'W') p1++;

else if (a[i] == 'L') p2++;

if (p1 >= 11 && (p1 - p2 >= 2) || p2 >= 11 && (p2 - p1 >= 2)) {

printf("%d:%d\n", p1, p2);

p1 = 0;

p2 = 0;

}

}

printf("%d:%d\n\n", p1, p2);

p1 = 0;

p2 = 0;

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

if (a[i] == 'W') p1++;

else if (a[i] == 'L') p2++;

if (p1 >= 21 && (p1 - p2 >= 2) || p2 >= 21 && (p2 - p1 >= 2)) {

printf("%d:%d\n", p1, p2);

p1 = 0;

p2 = 0;

}

}

printf("%d:%d\n", p1, p2);

return 0;

}

[P1176](https://client.vpn.nuist.edu.cn/https/webvpn893ff9021738b0357186c0f23fc2aed6e24ca283e886022bc5d861ea12f03963/course/94/submission) 小孩报数问题

#include<stdio.h>

#include<string.h>

#define MAX 64

char name[MAX][15];

int data[MAX];

int n;

void show(int current);

int main() {

int T, i = 0;

scanf("%d", &T);

getchar(); // 读取换行符

while (i < T) {

scanf("%s", name[i]);

i++;

}

int W, S;

scanf("%d", &W);

getchar(); // 读取换行符

scanf("%d", &S);

S--;

n = T;

for (i = 0; i < T; i++)

data[i] = i;

int current = W - 1;

while (n != 1) {

current = (current + S) % n;

show(data[current]);

for (int i = current; i < n - 1; i++)

data[i] = data[i + 1];

current = (current + 1) % n;

if (current != 0)

current--;

n--;

}

show(data[0]);

return 0;

}

void show(int current) {

printf("%s\n", name[current]);

}

# P1182 寻找能被3整除的斐波拿契数

#include <stdio.h>

// 计算斐波那契数列第 n 项模 3 的值

int fibonacciMod3(long long n) {

// 初始化斐波那契数列的前两项

int f0 = 7 % 3, f1 = 11 % 3;

// 如果 n 为 0 或 1，直接返回相应的值

if (n == 0) return f0;

if (n == 1) return f1;

// 计算第 n 项的值

for (long long i = 2; i <= n; i++) {

int temp = (f0 + f1) % 3;

f0 = f1;

f1 = temp;

}

return f1;

}

// 判断是否能被 3 整除的函数

char \*isDivisibleBy3(long long n) {

// 计算第 n 项模 3 的值

int fnMod3 = fibonacciMod3(n);

// 如果模 3 的值为 0，则返回 "yes"，否则返回 "no"

if (fnMod3 == 0) {

return "yes";

} else {

return "no";

}

}

int main() {

long long n;

// 读取输入并计算结果

while (scanf("%lld", &n) != EOF) {

printf("%s\n", isDivisibleBy3(n));

}

return 0;

}

[P1187](https://client.vpn.nuist.edu.cn/https/webvpn893ff9021738b0357186c0f23fc2aed6e24ca283e886022bc5d861ea12f03963/course/94/submission) 半素数

#include <stdio.h>

// Function to check if a number is prime

int isPrime(int n) {

if (n <= 1) return 0; // Numbers less than or equal to 1 are not prime

for (int i = 2; i \* i <= n; i++) {

if (n % i == 0) return 0; // If it has a divisor other than 1 and itself, it's not prime

}

return 1; // Otherwise, it's prime

}

// Function to check if a number is semi-prime

int isSemiPrime(int n) {

// Iterate through numbers from 2 to n/2 to find two prime factors

for (int i = 2; i <= n / 2; i++) {

if (n % i == 0) {

// If i is a prime number and n/i is also prime, then n is semi-prime

if (isPrime(i) && isPrime(n / i)) {

return 1; // Semi-prime

}

}

}

return 0; // Not semi-prime

}

int main() {

int N;

while (scanf("%d", &N) != EOF) {

// Check if N is semi-prime and print the result

if (isSemiPrime(N)) {

printf("Yes\n");

} else {

printf("No\n");

}

}

return 0;

}

[P1191](https://client.vpn.nuist.edu.cn/https/webvpn893ff9021738b0357186c0f23fc2aed6e24ca283e886022bc5d861ea12f03963/course/94/submission) Children’s Queue

#include <stdio.h>

int a[1001][300];

void fun(){

int i,j,t;

a[1][1]=1;

a[2][1]=2;

a[3][1]=4;

a[4][1]=7;

for(i=5;i<=1000;i++){

int pu=0;

for(j=1;j<300;j++){

t=a[i-4][j]+a[i-2][j]+a[i-1][j]+pu;

a[i][j]=t%10;

pu=t/10; //该位够十以后进到下一位

//此处还有一个作用，就是把没有数的都等于0

}

}

}

int main(void)

{

int n,i;

fun();

while(scanf("%d",&n)!=EOF){

i=299;

while(i>0&&!a[n][i]){ //i>0并且a[n][i]=0,i--是为了找到a[n]的最后一位

i--;

}

for(;i>0;i--)

printf("%d",a[n][i]);

printf("\n");

}

return 0;

}

# P1202 字符串插入

#include <stdio.h>

#include <string.h>

int main() {

char a[16]; // String a with maximum length of 15 characters

char b[4]; // String b with length of 3 characters

// Read input until end of file

while (scanf("%s %s", a, b) != EOF) {

int p = 0;

// Find the index of the character with maximum ASCII value in a

for (int i = 0; a[i] != '\0'; i++) {

if (a[i] > a[p]) {

p = i;

}

}

// Print the modified string by concatenating substrings

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

printf("%c", a[i]);

}

printf("%s", b);

for (int i = p + 1; a[i] != '\0'; i++) {

printf("%c", a[i]);

}

printf("\n");

}

return 0;

}

[P1213](https://client.vpn.nuist.edu.cn/https/webvpn893ff9021738b0357186c0f23fc2aed6e24ca283e886022bc5d861ea12f03963/course/94/submission) 开灯问题(1)

#include <stdio.h>

#include <stdbool.h>

int main() {

int n, k;

while (scanf("%d %d", &n, &k) != EOF) {

bool lights[1001] = {false}; // Array to represent the status of lights, initialized as all off

for (int i = 1; i <= k; i++) { // Loop through each person

for (int j = 1; j <= n; j++) { // Toggle switches for each multiple of i (i.e., person's number)

if (j % i == 0) {

lights[j] = !lights[j]; // Toggle the light

}

}

}

bool first = true;

for (int i = 1; i <= n; i++) { // Output the lights that are off

if (!lights[i]) {

if (!first) {

printf(" ");

}

printf("%d", i);

first = false;

}

}

printf("\n");

}

return 0;

}

[P1220](https://client.vpn.nuist.edu.cn/https/webvpn893ff9021738b0357186c0f23fc2aed6e24ca283e886022bc5d861ea12f03963/course/94/submission) 皇后摆放问题

#include<stdio.h>

char a[9][9],filled[9]={};

int dfs(int n){

if(n==8) return 1;

if(filled[n])return dfs(n+1);

int i,j,ans=0,can;

for(i=0;i<8;i++){

can=1;

for(j=0;j<8;j++){

if(j!=i && a[n][j]){

can=0;

break;

}

if(j!=n && a[j][i]){

can=0;

break;

}

}

for(j=1;n-j>-1&&i-j>-1;j++)

if(a[n-j][i-j]){

can=0;

break;

}

for(j=1;n+j<8&&i-j>-1;j++)

if(a[n+j][i-j]){

can=0;

break;

}

for(j=1;n-j>-1&&i+j<8;j++)

if(a[n-j][i+j]){

can=0;

break;

}

for(j=1;n+j<8&&i+j<8;j++)

if(a[n+j][i+j]){

can=0;

break;

}

if(can){

a[n][i]=1;

ans+=dfs(n+1);

a[n][i]=0;

}

}

return ans;

}

int main(){

int i,j;

for(i=0;i<8;i++)for(j=0;j<8;j++){

scanf("%d",&a[i][j]);

if(a[i][j])filled[i]=1;

}

printf("%d",dfs(0));

return 0;

}

[P1277](https://client.vpn.nuist.edu.cn/https/webvpn893ff9021738b0357186c0f23fc2aed6e24ca283e886022bc5d861ea12f03963/course/94/submission) Speak Time

#include<stdio.h>

int hours(int h)

{

switch(h)

{

case 0: printf("zero o'clock");break;

case 1: printf("one o'clock");break;

case 2: printf("two o'clock");break;

case 3: printf("three o'clock");break;

case 4: printf("four o'clock");break;

case 5: printf("five o'clock");break;

case 6: printf("six o'clock");break;

case 7: printf("seven o'clock");break;

case 8: printf("eight o'clock");break;

case 9: printf("nine o'clock");break;

case 10: printf("ten o'clock");break;

case 11: printf("eleven o'clock");break;

case 12: printf("twelve o'clock");break;

case 13: printf("thirteen o'clock");break;

case 14: printf("fourteen o'clock");break;

case 15: printf("fifteen o'clock");break;

case 16: printf("sixteen o'clock");break;

case 17: printf("seventeen o'clock");break;

case 18: printf("eighteen o'clock");break;

case 19: printf("nineteen o'clock");break;

case 20: printf("twenty one o'clock");break;

case 21: printf("twenty two o'clock");break;

case 22: printf("twenty three o'clock");break;

case 23: printf("twenty three o'clock");break;

}

}

int minutes(int m)

{

if(m>0&&m<=9)

switch(m)

{

case 0: printf("zero");break;

case 1: printf("one");break;

case 2: printf("two");break;

case 3: printf("three");break;

case 4: printf("four");break;

case 5: printf("five");break;

case 6: printf("six");break;

case 7: printf("seven");break;

case 8: printf("eight");break;

case 9: printf("nine");break;

}

if(m>=10&&m<=20)

switch(m)

{

case 10: printf("ten");break;

case 11: printf("eleven");break;

case 12: printf("twelve");break;

case 13: printf("thirteen");break;

case 14: printf("fourteen");break;

case 15: printf("fifteen");break;

case 16: printf("sixteen");break;

case 17: printf("seventeen");break;

case 18: printf("eighteen");break;

case 19: printf("nineteen");break;

case 20: printf("twenty");break;

}

if(m>20&&m<60)

switch(m/10)

{

case 2: printf("twenty ");minutes(m%10);break;

case 3: printf("thirty ");minutes(m%10);break;

case 4: printf("forty ");minutes(m%10);break;

case 5: printf("fifty ");minutes(m%10);break;

}

}

int hm(int h)

{

switch(h)

{

case 0: printf("zero ");break;

case 1: printf("one ");break;

case 2: printf("two ");break;

case 3: printf("three ");break;

case 4: printf("four ");break;

case 5: printf("five ");break;

case 6: printf("six ");break;

case 7: printf("seven ");break;

case 8: printf("eight ");break;

case 9: printf("nine ");break;

case 10: printf("ten ");break;

case 11: printf("eleven ");break;

case 12: printf("twelve ");break;

case 13: printf("thirteen ");break;

case 14: printf("fourteen ");break;

case 15: printf("fifteen ");break;

case 16: printf("sixteen ");break;

case 17: printf("seventeen ");break;

case 18: printf("eighteen ");break;

case 19: printf("nineteen ");break;

case 20: printf("twenty ");break;

case 21: printf("twenty one ");break;

case 22: printf("twenty two ");break;

case 23: printf("twenty three ");break;

}

}

int main()

{

int h,m;

while(scanf("%d %d",&h,&m) != EOF) {

if(m==0)

hours(h);

else

{

hm(h);

minutes(m);

}

printf("\n");

}

return 0;

}

[P1279](https://client.vpn.nuist.edu.cn/https/webvpn893ff9021738b0357186c0f23fc2aed6e24ca283e886022bc5d861ea12f03963/course/94/submission) 晕头的蚂蚁

#include <stdio.h>

#define MAX\_N 100 // 最大测试数据组数

#define MAX\_ANT 100 // 最大蚂蚁个数

int main() {

int T; // 测试数据组数

scanf("%d", &T);

for (int t = 0; t < T; t++) {

int L, n; // 杆子长度和蚂蚁个数

scanf("%d %d", &L, &n);

int x[MAX\_ANT]; // 蚂蚁位置数组

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

scanf("%d", &x[i]);

}

int min = 0, max = 0;

int minX, maxX;

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

minX = x[i] < L - x[i] ? x[i] : L - x[i];

min = minX > min ? minX : min; // 最短时间

maxX = x[i] > (L - x[i]) ? x[i] : L - x[i];

max = maxX > max ? maxX : max; // 最长时间

}

printf("%d %d\n", min, max);

}

return 0;

}

[P1323](https://client.vpn.nuist.edu.cn/https/webvpn893ff9021738b0357186c0f23fc2aed6e24ca283e886022bc5d861ea12f03963/course/94/submission) 青蛙过河

#include <stdio.h>

int climbStairs(int n) {

int p = 0, q = 0, r = 1;

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

p = q;

q = r;

r = p + q;

}

return r;

}

int main() {

int n;

while(scanf("%d", &n)!= EOF) {

printf("%d\n", climbStairs(n) );

}

return 0;

}

[P1325](https://client.vpn.nuist.edu.cn/https/webvpn893ff9021738b0357186c0f23fc2aed6e24ca283e886022bc5d861ea12f03963/course/94/submission) 区间合并

#include <stdio.h>

#include <stdlib.h>

// 区间结构体定义

typedef struct Interval {

int start;

int end;

} Interval;

// 比较函数，用于排序

int compare(const void \*a, const void \*b) {

return ((Interval \*)a)->start - ((Interval \*)b)->start;

}

// 合并区间

void mergeIntervals(Interval \*intervals, int n) {

// 如果区间数量为 0 或 1，无需合并

if (n <= 1) return;

// 对区间按照左端点进行排序

qsort(intervals, n, sizeof(Interval), compare);

// 合并区间

int index = 0;

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

// 如果当前区间的左端点大于等于前一个区间的右端点，说明两个区间不相交，直接跳过

if (intervals[i].start > intervals[index].end) {

intervals[++index] = intervals[i];

}

// 如果当前区间的左端点小于等于前一个区间的右端点，说明两个区间相交或重叠，需要合并

else if (intervals[i].end > intervals[index].end) {

intervals[index].end = intervals[i].end;

}

}

// 输出合并后的区间

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

printf("%d %d\n", intervals[i].start, intervals[i].end);

}

}

int main() {

int n;

while (scanf("%d", &n) != EOF) {

Interval intervals[n];

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

scanf("%d %d", &intervals[i].start, &intervals[i].end);

}

mergeIntervals(intervals, n);

printf("\n");

}

return 0;

}

[P1531](https://client.vpn.nuist.edu.cn/https/webvpn893ff9021738b0357186c0f23fc2aed6e24ca283e886022bc5d861ea12f03963/course/94/submission) 字符串排序

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_N 10

#define MAX\_LENGTH 101

// 比较函数，用于字符串排序

int compare(const void \*a, const void \*b) {

return strcmp(\*(const char \*\*)a, \*(const char \*\*)b);

}

int main() {

int n;

scanf("%d\n", &n); // 注意这里多了一个换行符的读取

char strings[MAX\_N][MAX\_LENGTH]; // 存储输入的字符串

char \*pointers[MAX\_N]; // 存储字符串的指针

// 逐行读取输入的字符串

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

fgets(strings[i], MAX\_LENGTH, stdin);

// 去除字符串末尾的换行符

strings[i][strcspn(strings[i], "\n")] = '\0';

pointers[i] = strings[i];

}

// 对字符串按字典序升序排序

qsort(pointers, n, sizeof(char \*), compare);

// 输出排序后的字符串

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

printf("%s\n", pointers[i]);

}

return 0;

}

[P1688](https://client.vpn.nuist.edu.cn/https/webvpn893ff9021738b0357186c0f23fc2aed6e24ca283e886022bc5d861ea12f03963/course/94/submission) 餐厅点餐

#include <stdio.h>

int main() {

int t, minm, maxm;

scanf("%d", &t);

while (t--) {

int a[11], b[11], c[11], d[11], e[11];

int a1, b1, c1, d1, e1;

int num = 0, sum1 = 0, sum2 = 0, sum3 = 0;

scanf("%d %d %d %d %d", &a1, &b1, &c1, &d1, &e1);

for (int i = 0; i < a1; ++i) scanf("%d", &a[i]);

for (int i = 0; i < b1; ++i) scanf("%d", &b[i]);

for (int i = 0; i < c1; ++i) scanf("%d", &c[i]);

for (int i = 0; i < d1; ++i) scanf("%d", &d[i]);

for (int i = 0; i < e1; ++i) scanf("%d", &e[i]);

scanf("%d %d", &minm, &maxm);

for (int n4 = 0; n4 < d1; ++n4) {

for (int k4 = n4; k4 < d1; ++k4) {

if (k4 == n4) sum1 = d[n4];

else sum1 = d[n4] + d[k4];

for (int n5 = 0; n5 < e1; ++n5) {

for (int k5 = n5; k5 < e1; ++k5) {

if (k5 == n5) sum2 = e[n5];

else sum2 = e[n5] + e[k5];

for (int n1 = 0; n1 < a1; ++n1) {

for (int n2 = 0; n2 < b1; ++n2) {

sum3 = a[n1] + b[n2];

int sum = 0;

sum += sum3 + sum1 + sum2;

if (sum >= minm && sum <= maxm) num++;

}

}

for (int n3 = 0; n3 < c1; ++n3) {

int sum = 0;

sum += sum1 + sum2 + c[n3];

if (sum >= minm && sum <= maxm) num++;

}

}

}

}

}

printf("%d\n", num);

}

return 0;

}

[P1712](https://client.vpn.nuist.edu.cn/https/webvpn893ff9021738b0357186c0f23fc2aed6e24ca283e886022bc5d861ea12f03963/course/94/submission) 串联数字

#include <stdio.h>

#define MAX 100

int dp[MAX][MAX];

int Comp(const int \*a, const int \*b) {

return (\*a > \*b) ? 1 : ((\*a < \*b) ? -1 : 0);

}

int findFloor(int a);

int main() {

int n;

while (scanf("%d", &n) != EOF) {

int array[MAX];

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

scanf("%d", &array[i]);

int floor = findFloor(array[i]);

int k = array[i];

k -= floor \* (floor - 1) / 2;

k = k % 9;

if (k == 0) {

printf("9\n");

} else {

printf("%d\n", k);

}

}

}

return 0;

}

int findFloor(int k) {

int n = 1;

while (1) {

if (n \* (n + 1) >= 2 \* k) {

return n;

}

n++;

}

}

众所周知，插队在南京地铁里每天都在上演，当好市民在排队区按照顺序排队时，拟人生物会无视排队规则，直接插队到最前面。

假设有 *n*(0<*n*<100000) 个车门，在第 *i* 个车门处的排队区里有 *mi*​(0<*i*<=*n*,0<*mi*​<100000) 个好公民，且总的好公民个数不超过 100000(∑*i*=0*n*​*mi*​<100000)。

接下来会出现 *k*(0<*k*<100000) 个拟人生物依次到来并插队，第 *i* 个生物会插队到编号为 *di*​(0<*di*​<=*n*) 的车门队伍中并站在队伍的首位置。

每个好市民和拟人生物都有一个唯一的身份编号 (int 范围整数), 要求你在所有拟人生物插队完毕后，依次输出每个车门处上车时的身份编号顺序。

#### 输入描述

第一行输入 *n* 表示车门的个数。

第二行输入 *n* 个数，表示每个车门处好公民的排队人数，用空格隔开。

接下来 *n* 行，每行输入*mi*​ 个当前队伍中好市民的身份编号，用空格隔开。

第 *n*+3 行输入插队的拟人生物数量*k*

接下来*k* 行每行有两个整数，分别为插队生物的编号和其选择插队的车门编号，用空格隔开。

#### 输出描述

输出一共 *n* 行，第 *i* 行按照上车顺序输出车门 *i* 处上车的人员编号，编号间用空格隔开。

#include <stdio.h>

#include <stdlib.h>

#define MAX\_N 100000

#define MAX\_M 100000

int main() {

int size;

scanf("%d", &size);

int doors[size];

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

scanf("%d", &doors[i]);

}

int map[size][MAX\_M];

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

int num = doors[i];

for (int j = 0; j < num; j++) {

scanf("%d", &map[i][j]);

}

}

int other;

scanf("%d", &other);

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

int id, key;

scanf("%d %d", &id, &key);

for (int j = doors[key - 1]; j > 0; j--) {

map[key - 1][j] = map[key - 1][j - 1];

}

map[key - 1][0] = id;

doors[key - 1]++;

}

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

for (int j = 0; j < doors[i]; j++) {

printf("%d", map[i][j]);

if (j != doors[i] - 1) {

printf(" ");

}

}

printf("\n");

}

return 0;

}

[P1969](https://client.vpn.nuist.edu.cn/https/webvpn893ff9021738b0357186c0f23fc2aed6e24ca283e886022bc5d861ea12f03963/course/94/submission) 随机填表

p = (int \*)malloc(n \* n \* sizeof(int)); // (1) 填空

if(p == NULL)

{

printf("内存不足，请重新输入\n");

continue;

}

// (2) 填空

p = (int \*)malloc(n \* n \* sizeof(int));

// (3) 填空

\*(p + c) = i;

// (4) 填空

\*(p + i - 1)

**完整代码**

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<malloc.h>

const int mod = 46337;

int id = 1;

int myrand()

{

id += 2024;

id %= mod;

id = id \* id;

id %= mod;

return id;

}

int main()

{

int n, m, c;

int \*p;

while(1)

{

printf("输入n和m的值，以逗号隔开。例如4,9\n");

scanf("%d,%d", &n, &m);

if(n \* n < m)

{

printf("m值过大,请重新输入\n");

continue;

}

// 申请动态内存

p = (int \*)malloc(n \* n \* sizeof(int));

// 检查内存是否分配成功

if(p == NULL)

{

printf("内存不足，请重新输入\n");

continue;

}

if(p != NULL) break;

}

// 初始化矩阵

for(int i = 0; i < n \* n; i++)

\*(p + i) = 0;

for(int i = 1; i <= m; i++)

{

do{

c = myrand() % (n \* n);

} while(\*(p + c) != 0);

\*(p + c) = i;

}

// 打印矩阵

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

{

for(int j = 0; j < n; j++)

printf("%4d", \*(p + i \* n + j));

printf("\n");

}

free(p);

return 0;

}

[P1970](https://client.vpn.nuist.edu.cn/https/webvpn893ff9021738b0357186c0f23fc2aed6e24ca283e886022bc5d861ea12f03963/course/94/submission) 质数合数链表

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<malloc.h>

const int mod = 46337; // 用于生成伪随机数的模数

int id = 1; // 全局变量，用于生成伪随机数

// 生成伪随机数的函数

int myrand(int x)

{

id += 2024; // 增加一个固定值

id %= mod; // 取模

id += x;

id %= mod;

id = id \* id; // 平方

id %= mod; // 取模

id ++;

id %= mod;

while (id % 2 == 0 || id % 3 == 0)

{

id += 5; // 增加一个固定值

id %= mod; // 取模

}

while(id == 1){

id ++;

id %= mod;

while (id % 2 == 0 || id % 3 == 0)

{

id += 5; // 增加一个固定值

id %= mod; // 取模

}

}

return id % mod; // 返回伪随机数

}

// 链表节点的结构体定义

struct NODE

{

int data; // 节点中的数据

struct NODE \* pnext; // 指向下一个节点的指针

};

typedef struct NODE Node;

// 将数据插入到链表中的函数

void insert\_to\_list(Node \* phead, int data)

{

Node\* p;

// 遍历链表直到尾部

for(p = phead; p->pnext; p = p->pnext);

// 创建新节点并插入数据

p->pnext = (Node \*)malloc(sizeof(Node));

p = p->pnext;

p->pnext = NULL;

p->data = data;

}

int main()

{

Node a, b, \*p; // 声明两个链表节点和一个指针

int n, i, d, j; // 声明变量用于控制循环和存储随机数

a.pnext = b.pnext = NULL; // 初始化链表头指针为NULL

scanf("%d", &n); // 获取用户输入的整数n，表示随机数的个数

// 生成n个随机数，并根据是否为素数将其插入到两个链表中

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

{

d = myrand(n); // 调用myrand()函数生成伪随机数

// 判断随机数是否为素数

int f = 0;

for(int j = 2; j < d; j++)

if(d % j == 0){

f = 1;

break;

}

if(f)

insert\_to\_list(&b, d); // 是合数，插入到链表b中

else

insert\_to\_list(&a, d); // 是素数，插入到链表a中

}

// 打印链表a的内容

printf("list 1: \n");

for(p = a.pnext; p; p = p->pnext)

printf("%d\t", p->data);

// 打印链表b的内容

printf("\nlist 2: \n");

for(p = b.pnext; p; p = p->pnext)

printf("%d\t", p->data);

return 0;

}

[P1226](https://client.vpn.nuist.edu.cn/https/webvpn893ff9021738b0357186c0f23fc2aed6e24ca283e886022bc5d861ea12f03963/course/94/submission) 数字合并

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_LENGTH 100

// 比较函数，按照特定规则比较两个字符串的大小

int compare(const char \*a, const char \*b) {

char concat1[MAX\_LENGTH \* 2], concat2[MAX\_LENGTH \* 2];

strcpy(concat1, a);

strcat(concat1, b);

strcpy(concat2, b);

strcat(concat2, a);

return strcmp(concat1, concat2);

}

// 冒泡排序函数，对字符串数组进行排序

void bubbleSort(char \*\*array, int n) {

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (compare(array[j], array[j + 1]) < 0) {

char \*temp = array[j];

array[j] = array[j + 1];

array[j + 1] = temp;

}

}

}

}

int main() {

int n;

while (scanf("%d", &n) == 1) {

char \*\*array = (char \*\*)malloc(n \* sizeof(char \*));

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

array[i] = (char \*)malloc(MAX\_LENGTH \* sizeof(char));

scanf("%s", array[i]);

}

// 排序

bubbleSort(array, n);

// 输出排序后的结果

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

printf("%s", array[i]);

}

printf("\n");

// 释放内存

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

free(array[i]);

}

free(array);

}

return 0;

}

#### 题目描述

食堂里正在举办买卤肉饭抽奖的活动，因为奖品是精美手办，所以吸引了大量的二次元，排起了长队。

这个队伍有一个奇特之处，进入队伍的方式不是从队伍的尾部进入，而是用一个变量 *i* 表示进入队伍的位置，如果 *i* 处进入了一个新的学生，那么从 1到 *i*−1 的排队学生不变，而原本在 *i* 位及其后方的学生需要向后挪动一位，空出的第 *i* 位由新来的学生占据。同理，如果第 *i* 位的学生离开，后方的学生会全部向前挪动一位。

一共有三种操作：

1. 变量 *i* 加一，指令格式为：1。
2. 在 *i* 位置加入一名学生，指令格式为：2 *x*, *x* 是进入队伍的学生编号，保证不重复。
3. 在 *i* 位置离开一名学生，指令格式为：3。

队伍位置编号从 1 开始，初始 *i* 的值为 1，一共给出 *n*(1<=*n*<=1000) 个操作，全部操作完成之后，依次输出队伍里的学生编号。输入会保证操作总是合法的，*i* 保证不会大于当前队伍里学生数 + 1, 且不会在没有学生的位置出现学生离开的操作。

#### 输入描述

第一行输入正整数 *n*，表示操作次数。

接下来的 *n* 行每行为一个操作指令。

#### 输出描述

输出一行，为队伍里从前到后的学生编号，用空格隔开。

[P1110](https://client.vpn.nuist.edu.cn/https/webvpn893ff9021738b0357186c0f23fc2aed6e24ca283e886022bc5d861ea12f03963/course/94/submission) 校门外的树

#### 题目描述

某校大门外长度为 *L* 的马路上有一排树，每两棵相邻的树之间的间隔都是 11 米。我们可以把马路看成一个数轴，马路的一端在数轴 0 的位置，另一端在 *L* 的位置；数轴上的每个整数点，即 0,1,2,…,*L* , 都种有一棵树。由于马路上有一些区域要用来建地铁。这些区域用它们在数轴上的起始点和终止点表示。已知任一区域的起始点和终止点的坐标都是整数，区域之间可能有重合的部分。现在要把这些区域中的树 (包括区域端点处的两棵树) 移走。你的任务是计算将这些树都移走后，马路上还有多少棵树.

#### 输入描述

第一行输入两个整数 *L*(1≤*L*≤100000) 和*M*(1≤*M*≤100) , *L* 代表马路的长度，*M* 代表区域的数目，*L* 和 *M* 之间用一个空格隔开。接下来的 *M* 行每行包含两个不同的整数，用一个空格隔开，表示一个区域的起始点和终止点的坐标.

#### 输出描述

占一行，马路上剩余的树的数目

#include <stdio.h>

int main() {

int L, M;

while (scanf("%d %d", &L, &M) != EOF) {

// 初始化马路上的树

int road[L + 1];

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

road[i] = 1; // 初始都有树

}

// 移走区域内的树

int start, end;

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

scanf("%d %d", &start, &end);

for (int j = start; j <= end; j++) {

road[j] = 0; // 标记该位置没有树

}

}

// 统计剩余的树的数量

int remaining\_trees = 0;

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

if (road[i] == 1) {

remaining\_trees++;

}

}

// 输出结果

printf("%d\n", remaining\_trees);

}

return 0;

}

# P1183 Image Transformation

#include <stdio.h>

#include <stdlib.h>

int main() {

int m, n;

int case\_num = 0;

while (1) {

if (scanf("%d %d", &m, &n) != 2 || (m == 0 && n == 0)) {

break;

}

int \*r\_v = (int\*)malloc(m \* n \* sizeof(int));

int \*b\_v = (int\*)malloc(m \* n \* sizeof(int));

int \*g\_v = (int\*)malloc(m \* n \* sizeof(int));

if (r\_v == NULL || b\_v == NULL || g\_v == NULL) {

perror("Memory allocation failed");

return 1;

}

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

scanf("%d", &r\_v[i]);

}

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

scanf("%d", &b\_v[i]);

}

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

scanf("%d", &g\_v[i]);

}

printf("Case %d:\n", ++case\_num);

for (int j = 0; j < m \* n; j++) {

printf("%d", (r\_v[j] + b\_v[j] + g\_v[j]) / 3);

if ((j + 1) % n == 0) {

printf("\n");

} else {

printf(",");

}

}

free(r\_v);

free(b\_v);

free(g\_v);

}

return 0;

}