中国人民大学
RENMIN UNIVERSITY OF CHINA

C程序设计

期末复习专题

（往年考题分析）

余力

buaayuli@ruc.edu.cn

2021-1 四平方和

四平方和定理：每个正整数均可表示为四个非负整数(可以是0)的平方和，即对于任意一个正整数n，总有四个非负整数a、b、c、d，使得$n=a^2+b^2+c^2+d^2$。现给定n，求字典序最小的一组解。

提示：从小到大枚举问题的可能a、b、c、d，按照数量关系要求进行判断，较小的一组解即为本题要求的解。

输入：一个整数n，0<n<5000000。

输出：4个非负整数a b c d，中间以空格分隔，末尾没有空格直接换行。

输入样例 1： 5

输出样例 1： 0 0 1 2

2021-2 疫情期间排座位

疫情期间，学校的考场需要防止同学因位置太密集而发生交叉感染，设计了一种座位排列规则。每一个教室都是规则的N*N的方形，每个教室有N名同学参加考试，必须使任意两名同学，不在同一行，不在同一列，也不在同一条正对角线或反对角线上，你需要判断给出的座位分布顺序是否符合该要求。

输入：第一个数字，为教室座位的行数或列数N，即教室有N*N座位。

其余N个数字，代表着N个同学座位的分布，即第i个元素代表列数为i的同学所在座位的行数(缺省安排各个同学在不同的列上，比如第1个同学在第1列，第2个同学在第2列，...)。

注意：这里的行、列从1开始编号。

输出：判断座位安排是否符合防疫规定。

输入样例 1： 8 4 6 8 2 7 1 3 5

输出样例 1： YES

2021-3 四分位点与编码

寻找整数数列的上下四分位点，并且对数列进行编码。

四分位点的定义为，对数列进行从小到大的排序，落在总个数N的1/4的位置(N*1/4取整数)的元素为下四分位点Q1，落在总个数N的3/4的位置(N*3/4取整数)的元素为上四分位点Q3。

对原始输入数列进行编码，把小于Q1的数编码为C，把大于等于Q1且小于Q3的数编码为B，把大于等于Q3的数编码为A，进行输出。

输入：为一行，第一个数为个数N，后续是N个整数。

输出：为一行，对原数列的编码。

输入样例 1： 9 9 8 7 6 5 4 3 2 1

输出样例 1： AAABBBBCC

2021-4 隐藏的字符串

给定一个混乱无序的字符串S，字符串中不含空格，同时给定一个有N个字符串(即单词)的词典D，你需要在S中找出词典D中最长的字符串L，这个字符串L可以通过删除S中的一些字符得到。若可以找到多个符合要求的字符串L，则输出字典序最小的字符串。若找不到，则输出“No exist!”。

现在结合如下样例，进行输入输出说明。

输入：第一行为一个长的字符串bapcplea。

第二行的4，表示字典里有4个单词。后续每行输入一个单词。

输出：为apple，因为根据题目规则，长串bapcplea里面包含apple、ale、plea等，apple是最长的串，且字典序最小。

输入样例 1：

bapcplea ↵

4 ↵

ale ↵

apple ↵

monkey ↵

plea ↵

输出样例 1： apple ↵

2021-5 彩票统计

在山的那边、海的那边，有一群蓝精灵有一个不务正业、游手好闲、有点小钱还又梦想不依赖父母就能一夜暴富的纨绔子弟。他懒散惯了，于是想到能够实现暴富目标的唯一方法就是买彩票(危险动作，请勿模仿)。他买彩票没有什么特别的方法，每次都是随机买，而且买的数量特别多，后来他发现这样做有一个问题，就是开奖后很难找到自己中奖的那些彩票，兑奖很难。所以希望你帮忙做个程序，要求能按顺序统计他都买了哪些数字组合、每种组合买了多少张。这种彩票规则如下：每买一张需要从0~9中选择6个数字作为一个组合，其中数字可以重复，兑奖时不论数字顺序。因此，你的程序里每张彩票的6个数字需要按从小到大的顺序排好。另外，由于他思路比较清奇，最想知道的是那些买的多的数字组合有没有中大奖，所以他希望排序优先按照每种数字组合购买的数量，买的多的放在前面，买的少的放在后面，若数量一致，再按彩票从左到右的字典序排，即先比较第一个数字，如果相同再比较第二个，依此类推，直至遇到不同的数字，先给小的所在组合，再给出大的。

输入：共N+1行。

第1行为一个整数N，代表彩票的数量，N最大有25万；

第2至N+1行，每行包含6个整数，范围为0~9，以空格分隔，代表每张彩票里包含的数字组合。

输出：按照题目要求顺序输出多行，每行包含7个整数——前6个为排好的彩票的数字，第7个数表示这张彩票购买的数量。整数间以空格分隔，最后一个整数后换行。

输入样例 1：

```
4
5 2 3 3 1 3
4 2 5 2 3 5
0 9 8 0 7 7
1 2 5 3 3 3
```

输出样例 1：

```
1 2 3 3 3 5 2
0 0 7 7 8 9 1
2 2 3 4 5 5 1
```

```
int n; → scanf("%d", &n);
for (int i = 0; i < n; i++) {
    → for (int j = 0; j < 6; j++)
    → → scanf("%d", &nums[i][j]);
    → sort(nums[i], nums[i] + 6); → }
for (int i = 0; i < 6; i++)
    → lottery[0].numbers[i] = nums[0][i];
lottery[0].count = 1;
```

```
typedef struct {
    → int numbers[6];
    → int count;
} LOTTERY;
LOTTERY lottery[250000];
```

```
Int same(int a[], int b[]) {
    → int i = 0;
    → for (; i < 6; i++)
    → → if (a[i] != b[i]) → break;
    → if (i == 6) → return 1;
    → return 0;
}
```

```
int comp(const void *a1, const void *a2) {
    → LOTTERY *p1 = (LOTTERY *)a1;
    → LOTTERY *p2 = (LOTTERY *)a2;
    → if (p1->count > p2->count) → return -1;
    → else if (p1->count < p2->count) → return 1;
    → else
    → → for (int i = 0; i < 6; i++)
    → → → if (p1->numbers[i] < p2->numbers[i]) → → return -1;
    → → → else if (p1->numbers[i] > p2->numbers[i]) → return 1;
}
```

```
qsort(lottery, group, sizeof(LOTTERY), comp);
for (int i = 0; i < group; i++) {
    for (int j = 0; j < 6; j++)
        printf("%d ", lottery[i].numbers[j]);
    printf("%d\n", lottery[i].count);
}

int group = 1;
for (int i = 1; i < n; i++) {
    int j = 0;
    for (; j < group; j++)
        if (same(lottery[j].numbers, nums[i]))
            {lottery[j].count++; break;}
    if (j == group) {
        for (int k = 0; k < 6; k++)
            lottery[group].numbers[k] = nums[i][k];
        lottery[group].count = 1;
        group++;    }
}
```

2021-6 结对编程

有两个学习小组，分别是小组A和小组B，人数一样。请给小组A和小组B结对子，让他们一起讨论，互换角色进行编码、调试和Review，共同进步。
输入：为3行，第一行为小组A或者小组B的人数N，两个小组的人数一样，人员编号为0到N-1。
第2行为“必须在一起”的约束，第一个整数表示有几个约束条件，后续是各个约束，每个约束为“M空格N”的形式，表示小组A的下标M者和小组B的下标N者必须在一起。
第3行为“不能在一起”的约束，第一个整数表示有几个约束条件，后续是各个约束，每个约束为“M空格N”的形式，表示小组A的下标M者和小组B的下标N者不能在一起。
输出：为若干配对方案，每个换行。有多个配对方案，则按照字典序输出。
每个配对方案的形式为一系列配对，每个配对的形式为“(M,N)”。配对方案内部按照从小编号到大编号的顺序排列。
没有任何配对方案，输出NONE。

输入样例 2:

5
2 0 0 1 1
2 2 3 2 4

输出样例 2:

(0,0)(1,1)(2,2)(3,3)(4,4)
(0,0)(1,1)(2,2)(3,4)(4,3)

```
void Try(int i){
    if (i == num){
        if (IsOK(peidui))
            for (int j = 0; j < num; j++)
                printf("(%d,%d)", j, peidui[j]);
        printf("\n");
    }else
        for (int j = 0; j < num; j++)
            if (used[j] == 0){
                peidui[i] = j;
                used[j]=1;
                Try(i+1);
                used[j] = 0;    }
}
```

```
int IsOK(int peidui[])
{   for (int k = 0; k < num; k++){
        for (int j = 0; j < shouldn; j++)
            if (k==sdna[j] && peidui[k]==sdnb[j]) r
        for (int j = 0; j < should; j++)
            if ((k==sda[j] && peidui[k]!=sdb[j]) || (k!=sda[j] && peidui[k]==sdb[j])) return 0; }
    return 1;
}
```

```
int num, should, shouldn;
int peidui[100], used[100]={0};
int sda[100], sdb[100], sdna[100], sdnb[100];

int main(){
    scanf("%d", &num);
    scanf("%d", &should);
    for (int i = 0; i < should; i++){
        scanf("%d", &sda[i]);
        scanf("%d", &sdb[i]); }
    scanf("%d", &shouldn);
    for (int i = 0; i < shouldn; i++){
        scanf("%d", &sdna[i]);
        scanf("%d", &sdnb[i]);}
    Try(0);
    return 0;
}
```

2020-1 encrypt

给定一个仅包含小写字母的字符串str，对其进行加密操作。加密方法是对于str的每个字符，在字母表上向后按照一个固定数目n进行偏移（偏移超过字母z则从a开始继续计算），将偏移后得到的字符串输出即为加密后的字符串。

【输入格式】

输入两行。

第1行，字符串str，str中只包含小写字母。

第2行，正整数n，代表偏移量。

【输出格式】

输出一行，为加密后的字符串。

【输入样例1】

happynewyear

4

【输出样例1】

lettcriaciev

2020-2 Find

在一个 n * n 的二维数组中，每一行都按照从左到右递增的顺序排序，每一列都按照从上到下递增的顺序排序，请查找某个数字是否在这个二维数组中。例如，下面的数组：

【输入格式】

第1行包含1个整数n，表示二维矩阵行数和列数；

第2到n+1行，每行n个整数，每2个整数之间用一个空格隔开；

第n+2行是一个整数，表示待查找的数字的个数k；

第n+3行包含k个整数a1,a2,a3,...,ak，每2个整数之间用一个空格隔开，为k个需要查找的整数。

1	2	4	6
3	13	17	23
8	21	26	36
10	31	39	43

【输入样例 1】

```
4
1 2  4  6
3 13 17 23
8 21 26 36
10 31 39 43
3
8 65 21
```

【输出样例 1】

```
20
-1
21
```

2020-3 Address

IPv4是 Internet Protocol version 4 的缩写，表示IP协议的第四个版本。互联网上绝大多数的通信流量都是以IPv4数据包的格式封装的。IPv4使用32位2进制位的地址，因此大约只有43亿个地址。IPv4通常用点分十进制记法书写，例如192.168.0.1，其中的数字都是十进制的数字，中间用实心圆点分隔。有效的IPv4地址正好含四个整数（每个整数位于0到255之间组成，且不能含有前导0），整数之间用'.'分隔。给定一个只包含数字的字符串，请给出由这串数字可以得到的所有可能的 IP 地址，注意字符串中数字前后顺序不能变。如："0.1.2.201"和 "192.168.1.1" 是有效IP 地址，但是"0.011.255.245"、"192.168.1.312"和"192.168@1.1"是无效的IP地址。

【输入格式】

1行，包含1个字符串，字符串中只包含数字，字符串的长度不超过12。

【输出格式】

若干行，每行一个输入数字串能得到有效IP地址。按照字符串的字典序输出（注意：规定字符. 排在任意数字字符之前）。

【输入样例1】25525511135

【输出样例1】

255.255.11.135

255.255.111.35

2020-4 String

给定两个字符串 s1 和 s2，写一个函数来判断 s2 是否包含 s1 的排列。若存在则输出s2中第一次包含s1的某种排列的字符串；若不存在输出false。即s1的排列之一是s2 的子串，并且在s2中最先出现，则输出s1的这个排列。

【输入格式】2行。

第1行，字符串s1。

第2行，字符串s2。

【输出格式】

1行，若存在则输出符合条件的s1的某个排列，若不存则输出false

【输入样例1】

abc

ecdbacbooo

【输出样例1】bac

2020-5 Matrix

给定一个n * n的矩阵A，如图所示，当处于方格(i, j)上时，可以尝试往上、下、左、右四个方向移动，移动的最大步长为该方格上的数字A[i][j]（0 ≤ A[i][j]≤ n)），即可以移动1步，2步，…，A[i][j]步。若A[i][j]的值为0，则表示在方格(i, j)上不能进行任何移动。一个合法的移动需要确保移动后所处的位置依然在n * n的矩阵内。

例如，在方格(3, 2)时，可以尝试往上、下、左、右四个方向移动1个步长或者2个步长（A[3][2]的值为2）。往左移动1步到达方格(3, 1)，是合法的移动；往左移动2步则超过n * n的矩阵范围，是不合法的移动。往右移动2步到达方格(3, 4)，是合法的移动，A[3][4]的值为0，表明到达单元格(3, 4)后就不能进行任何的移动。

一条从方格(1, 1)到方格(n, n)的可行路径是指从方格(1, 1)出发，经过若干次合法的移动之后可以到达方格(n, n)，并且相同的方格只访问一次。在所有的可行路径中，移动次数最小的路径为最优路径。请你编写一个程序，从(1, 1)到(n, n)的所有可行路径中，选择一条最优路径，输出相应的移动次数。输入数据保证有解。

	1	2	3	4
1	2	3	3	1
2	1	1	1	1
3	1	2	2	0
4	1	2	0	1

题型分析

- **基础知识**
  - 输入输出
  - 数组：数组访问、数组排序
  - 字符串：复制、比较

- **主要题型**
  - 枚举
  - 统计排序
  - 递归

枚举

- 自己要归纳几个枚举的典型题目
- 简单枚举（教室排课、火柴棒等式、address）
- 有顺序枚举（加法表达式、四平方和、选择奖品）
- 枚举与搜索区别
  - 枚举是最终方案有两、三部分组成，可有两、三重循环来表达
  - 搜索是N种情况，无法用重循环表达

统计排序

- 定义好数组
- 正确输入存储
- 统计
- 学会结构体
- 正常要争取做对

```
scanf("%d", &n);
// 输入 n 位学生信息
for (i = 0; i < n; i++) {
    scanf("%d %d", &StuNo, &KemuNum);
    for (sum = 0, j = 0; j < KemuNum; j++) {
        scanf("%d", &tmp);
        sum = sum + tmp; }
    if (KemuNum >= 2) {
        id[count] = StuNo;
        aver[count] = sum * 1.0 / KemuNum;
        count++; }
}
```

① 输入

② 统计

```
for (i = 0; i < count - 1; i++)
    for (j = 0; j < count – i-1; j++)
        if ((fabs(aver[j] - aver[j + 1]) < 1e-7 && id[j] > id[j + 1]) || aver[j] < aver[j + 1]) {
            tmp = id[j]; id[j] = id[j + 1]; id[j + 1] = tmp;
            tmpf = aver[j]; aver[j] = aver[j + 1]; aver[j + 1] = tmpf; }
for (i = 0; i < count; i++)
    printf("%d %.2lf\n", id[i], int(aver[i] * 100) / 100.0 );
```

③ 排序

递归

- 理解典型套路
- 搜索、全排列
- 反复理解讲过的题型

```
void perm (int k) {
 if (k == n) {
      for (int i = 0; i < n; i ++)
        printf("%d ", ary[selected[i]]);
      printf("\n");    }
 else
      for (int i = 0; i < n; i ++) {
        if (used[i] == 1) continue;
        used[i] = 1;
        selected[k] = i;
        perm(k+1);
        used[i] = 0;
      }
}
```

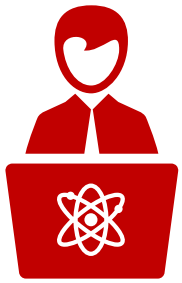
```
void Try(int x, int y, int len) {
      len ++;
      if (len > max_value)  max_value = len;

      if (a[x - 1][y] < a[x][y] && x - 1 > 0 )      Try(x - 1, y, len);
      if (a[x + 1][y] < a[x][y] && x + 1 <= c)     Try(x + 1, y, len );
      if (a[x][y - 1] < a[x][y] && y - 1 > 0 )      Try(x, y - 1, len);
      if (a[x][y + 1] < a[x][y] && y + 1 <= r)     Try(x, y + 1, len);
}
```

Try(i, j, 0)

备考注意事项

- **考前**
  - 考前积累准备好一些基础模块或函数，自己抄写在纸上
  - 记住一些典型类型题目套路，很多题目实际相似的
- **考时**
  - 仔细分析题意，确保看懂了题目
  - 先要确定题目类型（数值、字符串、枚举、递归）
  - 再定好大思路（先输入、再排序、再....），不要搞错方向
  - 编程时尽量函数模块化，分解难度，有的题目本来可能是综合
  - 实不会做，...?
- **考后**
  - 不管好坏，就不用想了

中国人民大学
RENMIN UNIVERSITY OF CHINA



谢谢大家！

感谢大家一学期来的支持！