ACM

国际大学生程序设计竞赛题解（2）

◎ 赵端阳　袁　鹤　编著

http: //www.phei.com.cn

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

ACM 国际大学生程序设计竞赛题解（2）

赵端阳　袁鹤　编著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

随着各大专院校参加 ACM/ICPC 热情的高涨，迫切需要有关介绍 ACM 国际大学生程序设计竞赛题解的书籍。本书根据浙江大学在线题库的部分题目，经过分类、筛选、汇编，并进行了解答（个别特别简单或者特别复杂的题目未选择），比较详细地分析和深入浅出地讲解了解题的方法和用到的算法。题目的类型包括基础编程、模拟、字符串处理、搜索、动态规划、回溯、图论、几何和数学题。

本书可以作为高等院校有关专业的本科和大专学生参加国际大学生程序设计竞赛的辅导教材，或者作为高等院校数据结构、C/C++程序设计或算法设计与分析等相关课程的教学参考书。

各题目的源代码及相关资料可在 http://www.hxedu.com.cn 下载。

图书在版编目（CIP）数据

ACM 国际大学生程序设计竞赛题解. 2 / 赵端阳，袁鹤编著. —北京：电子工业出版社，2010.7
ISBN 978-7-121-11171-6

Ⅰ. ①A… Ⅱ. ①赵… ②袁… Ⅲ. ①程序设计－竞赛－高等学校－解题 Ⅳ. ①TP311.1-44

中国版本图书馆 CIP 数据核字(2010)第 116153 号

责任编辑：陈晓莉
印　　刷：北京季蜂印刷有限公司
装　　订：三河市鹏成印业有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：787×1092　1/16　印张：22.25　字数：570 千字
印　　次：2010 年 7 月第 1 次印刷
印　　数：4000 册　　定价：39.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

ACM 国际大学生程序设计竞赛（ACM/ICPC：ACM International Collegiate Programming Contest）是由国际计算机界历史悠久、颇具权威性的组织 ACM 学会（Association for Computing Machinery，美国计算机协会）主办，是世界上公认的规模最大、水平最高的国际大学生程序设计竞赛，其目的旨在使大学生运用计算机来充分展示自己分析问题和解决问题的能力。1970 年在美国 Texas a&m 大学举办了首次区域竞赛，从而拉开了国际大学生程序设计竞赛的序幕。该项竞赛从 1970 年举办至今已历 29 届，因历届竞赛都荟萃了世界各大洲的精英，云集了计算机界的“希望之星”，而受到国际各知名大学的重视，并受到全世界各著名计算机公司的高度关注，成为世界各国大学生最具影响力的国际级计算机类的赛事。

此项赛事的主办目的不单是培养参赛选手的创造力，团队合作精神以及他们在软件程序开发过程中的创新意识，同时也是检测选手们在压力下进行开发活动的能力。因此，ACM 国际大学生程序设计竞赛是参赛选手展示计算机才华的广阔舞台，是著名大学计算机教育成果的直接体现，是信息企业与世界顶尖计算机人才对话的最好机会。该项竞赛分区域预赛和国际决赛两个阶段进行，各预赛区第一名自动获得参加世界决赛的资格，世界决赛安排在每年的 3～4 月举行，而区域预赛安排在上一年的 9～12 月在各大洲举行。从 1998 年开始，IBM 公司连续赞助该项赛事的世界决赛和区域预赛。这项比赛是以大学为单位组队（每支队伍由教练、3 名正式队员，一名后备队员组成）参赛。

国际 ACM 比赛 1996 年才进入中国内地，上海交通大学作为我国内地高校最早的参赛队之一，曾 7 次进军总决赛，并于 2002 年将 ACM 金杯首次带到亚洲。打破了几十年来欧美国家对这一赛事绝对的统治地位，更震惊了世界。

国际 ACM 比赛是世界上规模最大、历史最长、影响最深的全球性计算机专业竞赛，它要求每一名队员不仅具有扎实的数学功底、非凡的算法设计能力、娴熟的编程技巧，而且必须具备很好的协作精神、稳定的心理素质、快速的临场应变能力。

为了帮助各个大专院校的大学生们了解国际大学生程序设计竞赛，了解其程序设计的方法，提高参与校级、省级和亚洲区赛国际大学生程序设计竞赛的兴趣，特编写这本题解。

全书共分九章：

第 1 章，基础编程题。主要是比较容易的题目，也称简单题，尤其适合刚刚开始熟悉 ACM 大学生程序设计竞赛做题的同学。通过这些题目的练习，主要是熟悉数据的输入/输出格式，基本编程方法，在线提交系统的使用，常见错误及其对策。

第 2 章，模拟算法题。根据题目的要求逐步实现目标，虽然没有经典的算法可以使用，正确理解题目是关键的要素。例如题目“Robbery”和“Tournament Seeding”等。

第 3 章，字符串处理题。主要是字符串的处理，这也是考察参赛者细心的一类题目，需要对各种情况进行仔细的分析，予以正确的处理。例如题目“Language of FatmMouse”和“Bio-Informatics”等。

第 4 章，基本数据结构题。常见的有二叉树、堆栈和列表等，例如题目“Matrix Chain

Multiplication”和“Code the Tree”等。

第5章，搜索算法题。这是一个最常见的类型，通常有深度优先搜索和广度优先搜索算法。例如题目“Channel Allocation”和“Gamblers”等。

第6章，动态规划算法题。这些题目使用动态规划算法，会获得较快的运行时间和效率，例如题目“Monkey and Banana”和“FatMouse and Cheese”等。

第7章，回溯算法题。这些题目使用回溯算法，会获得较快的运行时间和效率，例如题目“Dreisam Equations”等。

第8章，图论算法题。主要包含拓朴排序、Floyd算法和二分图等，例如题目“Stockbroker Grapevine”和“Ouroboros Snake”等。

第9章，几何和数学题。主要是几何题和一些数学计算题，例如题目“Subway”和“Equidistance”等。

本部分通过大量的实例介绍了竞赛中常用的算法，并对如何灵活地应用这些算法进行了比较详细地分析和深入浅出地讲解。书中所用的语言是C/C++，并在MinGW Developer Studio中调试通过。在运行时间和内存占用的性能方面，C++并不比C语言优越多少，只是C++有丰富的模板库函数，输入/输出语句简单，编写的代码看起来格式更工整而已。在有大量输入/输出数据的题目中，书中会特别提醒读者需要使用C语言的输入/输出。显然，“Accepted”是我们的目标，算法是我们不断探索的道路，好的算法让我们容易达到目标。书中介绍的算法，虽然作者经过反复斟酌，不断优化和简化，但仍然不能认为是最优的。因为对算法的探索，正如金庸笔下的大侠们苦练武功一样，是没有止境的。

本书中虽然描述的是ACM国际大学生程序设计竞赛中最基本的算法，但它也已经超出了一般本科教科书所讲授的范围。清华大学计算机科学与技术系博士生导师、国际信息学奥林匹克中国队总教练吴文虎教授认为算法的确是艺术，“艺术与科学是相通的，都会给人以美的享受。”并感叹道：“体味思维艺术之美，我以为这可能是更高层次的享受”[1]。希望读者能从本书中体会到这一点。

本书可以作为高等院校有关专业的本科和大专学生参加国际大学生程序设计竞赛的辅导教材，或者作为高等院校数据结构、C/C++程序设计或算法设计与分析等相关课程的教学参考书，旨在培养和提高学生参加ACM国际大学生程序设计竞赛的兴趣。

本书在编写过程中得到了浙江大学在线裁判系统管理员的大力支持，在此表示非常感谢。

感谢沈仙桥同学翻译了大部分的题目。同时也感谢ACM参赛队员戚勇、林刚、王海江、沈懿华、郑彦良、戴俊、周智栋、应建伟等同学的热情帮助。

由于作者水平所限，书中难免有不足之处，恳请广大读者批评指正。作者电子邮件地址：727946579@qq.com和hzyuanhe@163.com。

作　者

2010年6月于杭州

[1] 吴文虎主编，孙贺编著，序，程序设计中的组合数学，清华大学出版社，2005年5月

目　录

第一章　基础编程题

在本章的题目大部分都比较简单，作为刚刚开始参加竞赛的练习题。

ZJU1086-Octal Fractions[1、2、3]

Time limit: 1 Seconds　　Memory limit: 32768KB

Fractions in octal (base 8) notation can be expressed exactly in decimal notation. For example, 0.75 in octal is 0.953125 (7/8+5/64) in decimal. All octal numbers of n digits to the right of the octal point can be expressed in no more than $3n$ decimal digits to the right of the decimal point.

Write a program to convert octal numerals between 0 and 1, inclusive, into equivalent decimal numerals. The input to your program will consist of octal numbers, one per line, to be converted. Each input number has the form $0.d_1d_2d_3 \cdots d_k$, where the d_i are octal digits (0..7). There is no limit on k. Your output will consist of a sequence of lines of the form

$0.d_1d_2d_3 \cdots d_k\ [8] = 0.D_1D_2D_3 \cdots D_m\ [10]$

where the left side is the input (in octal), and the right hand side the decimal (base 10) equivalent. There must be no trailing zeros, i.e. Dm is not equal to 0.

SAMPLE INPUT

```
0.75
0.0001
0.01234567
```

SAMPLE OUTPUT

```
0.75 [8] = 0.953125 [10]
0.0001 [8] = 0.000244140625 [10]
0.01234567 [8] = 0.020408093929290771484375 [10]
```

Problem Source:

South Africa 2001

【题目大意】

八进制表示的小数可用十进制数精确地表示。例如，八进制数 0.75 等于十进制数 0.953125

[1] http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemCode=1086

[2] http://acm.pku.edu.cn/JudgeOnline/problem?id=1131

[3] http://acmicpc-live-archive.uva.es/nuevoportal/data/problem.php?p=2245

（7/8+5/64）。八进制小数点右边的 n 位数可表示成小数位数不多于 $3n$ 位的十进制数。

编程任务：将 0～1 之间的八进制小数，转换为等值的十进制数。程序的输入是八进制数，每行一个，分别转换成十进制数。输入数据的格式为 $0.d_1d_2d_3\cdots d_k$，其中 d_i 为八进制数值（0…7），k 值没有限制。程序输出格式：

$0.d_1d_2d_3$ … d_k [8] = $0.D_1D_2D_3$ … D_m [10]

左边是输入的八进制数，而右边是等值的十进制数。小数后面没有拖零，也就是 D_m 不为 0。

【算法分析】

本题需要将八进制表示的小数用十进制数精确地表示，由于位数很多，需要采用高精度的方法计算。用数学公式表达这种转换关系：

$$(0.75)_8=(7\times8^{-1}+5\times8^{-2})_{10}=(0.93125)_{10}$$

被除数和除数都要使用高精度运算，编程是比较麻烦的。改变一下计算公式：

$$(0.75)_8=(7\times8^{-1}+5\times8^{-2})_{10}=((5/8+7)/8)_{10}=(0.93125)_{10}$$

也就是采用累除的方法，计算起来就很方便。从八进制小数的最低位开始，除以 8 后，与其前一位相加，一直到小数点后的第一位小数。如上例数据，只有两个小数位，分两步运算：

① 5/8=0.615

②（0.625+7）/8=0.93125

这样，我们只要实现除以 8 的高精度运算。

【程序代码】

程序名称：zju1086.c
题　　目：Octal Fractions
提交语言：C
运行时间：00:00.00
运行内存：388KB

```
#include <stdio.h>
#include <string.h>
#define MaxN  100
int main()
{
    char src[MaxN];                          //八进制的小数
    int i, j;
    while(scanf("%s",src) != EOF)
    {
        char dest[MaxN] = {'0'};             //十进制的小数
        int index = 0;                       //十进制小数的长度
        //从八进制小数的最后一位开始累除
        for(i=strlen(src)-1; i>1; i--)
        {
            int num = src[i] - '0';          //八进制小数的当前位
            //实现除以 8 的高精度运算
            int temp;
            //最后的余数 num 也必须除尽
```

```
            for(j=0; j<index || num; j++)
            {
                temp = num*10 + (j<index?dest[j]-'0':0);
                dest[j] = temp/8+'0';          //商
                num = temp%8;                  //余数
            }
            index = j;                         //十进制小数的实际位数
        }
        dest[j] ='\0';                         //字符串结束标志
        printf("%s [8] = 0.%s [10]\n", src, dest);
    }
    return 0;
}
```

ZJU1089-Lotto[1、2、3]

Time limit: 1 Seconds Memory limit: 32768KB

In a Lotto I have ever played, one has to select 6 numbers from the set {1,2,…,49}. A popular strategy to play Lotto — although it doesn't increase your chance of winning — is to select a subset S containing k ($k>6$) of these 49 numbers, and then play several games with choosing numbers only from S. For example, for $k=8$ and $S=\{1,2,3,5,8,13,21,34\}$ there are 28 possible games: [1,2,3,5,8,13], [1,2,3,5,8,21], [1,2,3,5,8,34], [1,2,3,5,13,21],…,[3,5,8,13,21,34].

Your job is to write a program that reads in the number k and the set S and then prints all possible games choosing numbers only from S.

Input Specification

The input file will contain one or more test cases. Each test case consists of one line containing several integers separated from each other by spaces. The first integer on the line will be the number k ($6<k<13$). Then k integers, specifying the set S, will follow in ascending order. Input will be terminated by a value of zero (0) for k.

Output Specification

For each test case, print all possible games, each game on one line. The numbers of each game have to be sorted in ascending order and separated from each other by exactly one space. The games themselves have to be sorted lexicographically, that means sorted by the lowest number first, then by the second lowest and so on, as demonstrated in the sample output below. The test cases have to be

[1] http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemCode=1089

[2] http://acm.pku.edu.cn/JudgeOnline/problem?id=2245

[3] http://acm.uva.es/p/v4/441.html

separated from each other by exactly one blank line. Do not put a blank line after the last test case.

Sample Input

```
7 1 2 3 4 5 6 7
8 1 2 3 5 8 13 21 34
0
```

Sample Output

```
1 2 3 4 5 6
1 2 3 4 5 7
1 2 3 4 6 7
1 2 3 5 6 7
1 2 4 5 6 7
1 3 4 5 6 7
2 3 4 5 6 7

1 2 3 5 8 13
1 2 3 5 8 21
1 2 3 5 8 34
1 2 3 5 13 21
1 2 3 5 13 34
1 2 3 5 21 34
1 2 3 8 13 21
1 2 3 8 13 34
1 2 3 8 21 34
1 2 3 13 21 34
1 2 5 8 13 21
1 2 5 8 13 34
1 2 5 8 21 34
1 2 5 13 21 34

1 2 8 13 21 34
1 3 5 8 13 21
1 3 5 8 13 34
1 3 5 8 21 34
1 3 5 13 21 34
1 3 8 13 21 34
1 5 8 13 21 34
2 3 5 8 13 21
2 3 5 8 13 34
2 3 5 8 21 34
2 3 5 13 21 34
2 3 8 13 21 34
2 5 8 13 21 34
3 5 8 13 21 34
```

Problem Source

University of Ulm Local Contest 1996

【题目大意】

在我曾经玩过的一种对号码的纸牌游戏（乐透）里，玩家必须从{1, 2,…, 49}中选 6 个数。玩 Lotto 的一个流行的策略是（虽然它并不增加你赢的机会）：就是从这 49 个数中，选出 k（$k>6$）个数组成一个子集 S，然后只从 S 里拿出牌来玩几局游戏。例如，$k=8$，$S=\{1, 2, 3, 5, 8, 13, 21, 34\}$，那么有 28 场可能的游戏：[1, 2, 3, 5, 8, 13], [1, 2, 3, 5, 8, 21], [1, 2, 3, 5, 8, 34], [1, 2, 3, 5, 13, 21],…,[3, 5, 8, 13, 21, 34]。

编程任务

读取数字 k 和一组数 S，输出由 S 中的数组成的所有可能的游戏。

输入格式

输入有一组或多组测试数据。每组测试数据一行，数与数之间用空格隔开。每行第一个整数是 k ($6<k<13$)，然后是从小到大排列的 k 个整数，即子集 S。当 k 为零时，输入结束。

输出格式

对于每组测试数据，输出所有可能的游戏，每个游戏占一行。游戏里的数字按升序排列，数与数之间有一个空格。所有游戏也要按字典序排列，也就是，首先比较各局游戏的第一个数的大小，然后是第二个数的大小，依次类推，如下面的输出样例所示。不同的测试例之间用一个空行隔开。最后一组测试例之后没有空行。

【算法分析】

本题看起来很复杂：它是要实现从一组数里挑选 6 个数的所有游戏组合，输出时每个游戏里的数字按升序排列，所有的游戏还要按字典序排列。所以在网上能够看到多种实现算法，如深度优先搜索等。

最简单的办法是模拟。因为数字已经有序，所以就从前往后挑选，这样挑选出来的数字，不仅每个游戏里的数字是升序的，而且所有游戏是字典序的。如图 1-1 所示，挑选第一个数字：

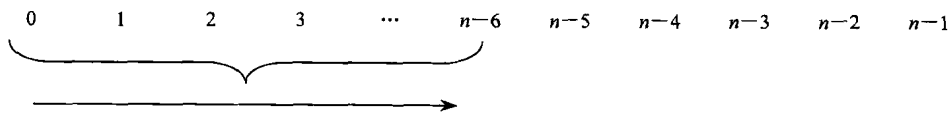

图 1-1　挑选第一个数字的示意图

依次在第 0～（$n-6$）个数字之间，挑选第一个数字，序号为 a。

第二个数字是第一个数字的序号 a 加 1，依次挑选数字，一直挑选到第 $n-5$ 个数字，序号为 b。依此类推。

【程序代码】

程序名称：	zju1089.c
题　　目：	Lotto
提交语言：	C++
运行时间：	00:00.00
运行内存：	392KB

```
#include<stdio.h>
int number[14];                          //存放原始数据

int main(){
    int line = 0;                        //空行标志
    int n;                               //数字的个数
    while(scanf("%d", &n) && n) {
        if (line) printf("\n");
        for(int i=0; i<n; ++i)           //读取子集 S
            scanf("%d", &number[i]);
        //模拟挑选数字，并输出结果
        for(int a = 0; a<n-5; ++a)
        for(int b=a+1; b<n-4; ++b)
        for(int c=b+1; c<n-3; ++c)
        for(int d=c+1; d<n-2; ++d)
        for(int e=d+1; e<n-1; ++e)
        for(int f=e+1;  f<n;  ++f)
            printf("%d %d %d %d %d %d\n",
                number[a],number[b],number[c],
                number[d],number[e],number[f]);
```

```
            line = 1;
        }
    }
```

ZJU1090-The Circumference of the Circle[1、2、3]

Time limit: 1 Seconds Memory limit: 32768KB

To calculate the circumference of a circle seems to be an easy task — provided you know its diameter. But what if you don't? You are given the cartesian coordinates of three non—collinear points in the plane.Your job is to calculate the circumference of the unique circle that intersects all three points.

Input Specification

The input file will conta in one or more test cases. Each test case consists of one line containing six real numbers $x_1, y_1, x_2, y_2, x_3, y_3$, representing the coordinates of the three points. The diameter of the circle determined by the three points will never exceed a million. Input is terminated by end of file.

Output Specification

For each test case, print one line containing one real number telling the circumference of the circle determined by the three points. The circumference is to be printed accurately rounded to two decimals. The value of π is approximately 3.141592653589793.

Sample Input

```
0.0 -0.5 0.5 0.0 0.0 0.5
0.0 0.0 0.0 1.0 1.0 1.0
5.0 5.0 5.0 7.0 4.0 6.0
0.0 0.0 -1.0 7.0 7.0 7.0
50.0 50.0 50.0 70.0 40.0 60.0
0.0 0.0 10.0 0.0 20.0 1.0
0.0 -500000.0 500000.0 0.0 0.0 500000.0
```

Sample Output

```
3.14
4.44
6.28
31.42
62.83
```

[1] http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemCode=1090

[2] http://acm.pku.edu.cn/JudgeOnline/problem?id=2242

[3] http://acm.uva.es/p/v4/438.html

```
632.24
3141592.65
```

Problem Source

University of Ulm Local Contest 1996

【题目大意】

假设你知道了圆的直径，计算其周长似乎是一件容易的事情。但是如果你又不会计算呢？在平面笛卡儿坐标系内，输入三个不在同一直线上的点。经过三个坐标点的圆只有一个，你的任务是求出该圆的周长。

输入格式

输入包含一组或多组测试数据。每个测试例一行，有 6 个实数 $x_1, y_1, x_2, y_2, x_3, y_3$，分别表示三个点的坐标。由这三个坐标点决定的圆直径不会超过 10^6。输入以文件结束符为标志。

输出格式

对于每组测试例，输出一行，是一个实数，由三个坐标点构成的圆周长。圆周长精确到百分位。π的近似值是 3.141592653589793。

【算法分析】

本题是已知三点的坐标，求外接圆的圆周长。

设圆心坐标是（x_0, y_0），半径是 r，则

$$(x-x_0)^2+(y-y_0)^2=r^2$$

将已知的三点坐标代入，求出半径 r，即可求出圆周长。但是解这个方程有点麻烦，利用三点构成的三角形，可以根据公式求解。

假设三角形的三边长分别是：a、b、c，对应角分别是 A、B、C，面积为 S，如图 1-2 所示。

（1）三角函数求解半径 r

根据余弦定理：

$$c^2=a^2+b^2-2ab\cos C$$

根据正弦定理：

$$\frac{c}{\sin C}=2r$$

$$\because \quad \sin^2 C+\cos^2 C=1$$

$$\therefore \quad \left(\frac{c}{2r}\right)^2+\left(\frac{a^2+b^2-c^2}{2ab}\right)^2=1$$

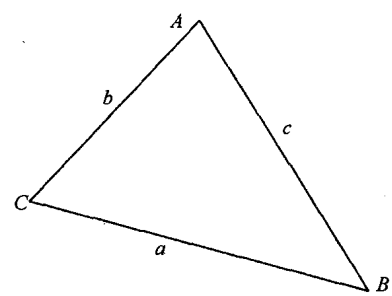

图 1-2　三角形的边角关系

从中解出 r 即可。

（2）面积公式求解半径 r

三角形的外接圆半径：

$$r=\frac{abc}{4S}$$

根据海伦公式计算面积：

$$S=\sqrt{p(p-a)(p-b)(p-c)}$$

其中：$p=$（a+b+c）/2。

【程序代码】

程序名称：　zju1090.c
题　　目：　The Circumference of the Circle
提交语言：　C
运行时间：　00:00.00
运行内存：　396K

```
#include <stdio.h>
#include <math.h>

#define PI 3.141592653589793
double square(double x){
    return x*x;
}

int main()
{
    double x1, y1, x2, y2, x3, y3;              //三点坐标
    double a2, b2, c2;                          //三边长的平方
    double r;                                   //半径
    while(scanf("%lf%lf%lf%lf%lf%lf", &x1,&y1,&x2,&y2,&x3,&y3)!=EOF)
    {
        //根据三角函数求解半径 r
        a2 = square(x1-x2) + square(y1-y2);
        b2 = square(x1-x3) + square(y1-y3);
        c2 = square(x3-x2) + square(y3-y2);
        r = sqrt(c2/(1-square(a2+b2-c2)/a2/b2/4))/2;
        printf("%.2lf\n", 2*PI*r);
    }
    return 0;
}
```

ZJU1095-Humble Numbers[1、2、3]

Time limit: 1 Seconds　Memory limit: 32768K

A number whose only prime factors are 2,3,5 or 7 is called a humble number. The sequence 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 15, 16, 18, 20, 21, 24, 25, 27, … shows the first 20 humble numbers. Write a program to find and print the *n* element in this sequence.

[1] http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemCode=1095
[2] http://acm.pku.edu.cn/JudgeOnline/problem?id=2247
[3] http://acm.uva.es/p/v4/443.html

Input Specification

The input consists of one or more test cases. Each test case consists of one integer n with $1 \leqslant n \leqslant 5842$. Input is terminated by a value of zero (0) for n.

Output Specification

For each test case, print one line saying "The nth humble number is number.". Depending on the value of n, the correct suffix "st", "nd", "rd", or "th" for the ordinal number nth has to be used like it is shown in the sample output.

Sample Input	Sample Output
1	The 1st humble number is 1.
2	The 2nd humble number is 2.
3	The 3rd humble number is 3.
4	The 4th humble number is 4.
11	The 11th humble number is 12.
12	The 12th humble number is 14.
13	The 13th humble number is 15.
21	The 21st humble number is 28.
22	The 22nd humble number is 30.
23	The 23rd humble number is 32.
100	The 100th humble number is 450.
1000	The 1000th humble number is 385875.
5842	The 5842nd humble number is 2000000000.
0	

Problem Source

University of Ulm Local Contest 1996

【题目大意】

质因数是 2, 3, 5 或者 7 的数称为丑数（humble number）。 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 15, 16, 18, 20, 21, 24, 25, 27,…是头 20 个 humble number。

写一个程序，输出上面序列中的第 n 个元素。

输入格式

输入一组或多组测试数据。每组测试数据一个整数 n（$1 \leqslant n \leqslant 5842$）。 输入 n 零（0）结束。

输出格式

对于每组测试数据,输出“The nth humble number is number.”。根据值 n，后缀有“st”,“nd”,“rd”或“th”，如下输出所示。

【算法分析】

本题是根据给定的数 n，计算相应的丑数（humble number）。在 University of Ulm Local