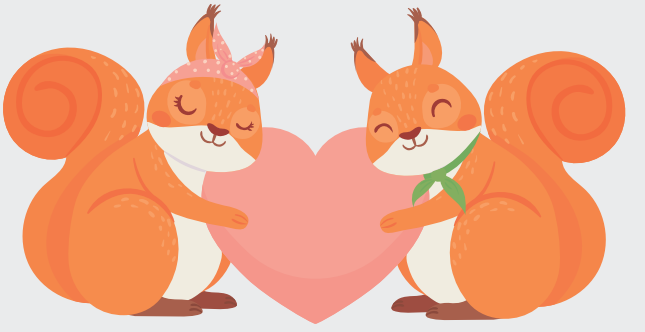
同学们从小就知道 1+1=2，那么请思考这样一个问题：1+1 会不会不等于 2？或者 1+1 在什么情况下等于 10（一零）？或许有同学想说，在算错的情况下等于 10。但是，这不是一个脑筋急转弯问题。其实，在特定条件下“1+1=10”这个算式是可以成立的，也就是当这个算式在二进制下是成立的，因为二进制是满 2 进 1。

有人说数学是枯燥的，其实不然，本章我们将一起学习一个非常有意思的“数字心灵感应”游戏。



1.1 问题情境

“数字心灵感应”是一个有趣的小游戏，游戏过程中有 N 个框，每个框里都有一些数字（同一数字可以在不同的框内出现），让玩家心里选中一个数字，并指出这个数字在哪些框内出现，就可以迅速地猜到玩家选中的数字是多少。

“数字心灵感应”小游戏设计如下：把 1~31 打乱分布在 5 个方框内，如图 1-1 所示，每个方框中有若干个 1~31 的数。我们发现，方框 A 里面的所有数都是奇数，方框 B 里面有奇数也有偶数，好像没有什么明显的规律，其他三个方框中的数也是有奇数也有偶数。不同的数出现的次数不同，有些数只出现在某一个方框内，有些数则出现在多个方框内。

A	B	C
1,3,5,7,9,11,13,15,17, 19,21,23,25,27,29,31	2,3,6,7,10,11,14,15,18, 19,22,23,26,27,30,31	8,9,10,11,12,13,14,15, 24,25,26,27,28,29,30,31

D	E
4,5,6,7,12,13,14,15,20, 21,22,23,28,29,30,31	16,17,18,19,20,21,22,23, 24,25,26,27,28,29,30,31

图 1-1 “数字心灵感应”小游戏

游戏的规则如下。

- 你心里想一个 1~31 的数，不要说出来。
- 我会问你几个简单问题，你想的数出现在 A 框吗？

B 框吗？ C 框吗？ D 框吗？ E 框吗？

- 在的话你就说在，不在的话你就说不在。
- 等你回答完了，我就可以在 1s 内感应出你心里想的那个数，保证百分之百正确，除非你撒谎。

这是如何做到的呢？有同学可能觉得是根据集合的原理，这个方框有那个方框也有，那就是相同的部分了，找到相同的部分就行了。这种方法是可以推测出你心里想的那个数字，但是它的速度并没有那么快。

“数字心灵感应”小游戏的设计有一个更为简便的方法，根本就不用看，只要你告诉他在哪几个方框里面出现就可以快速算出来。

所以，这个小游戏不是心灵感应，而是可以用小学一年级的加法算式计算出来的。

1.2 案例：数字心灵感应

其实，“数字心灵感应”游戏的奥妙在于十进制与二进制之间的转换，每个十进制整数转换成二进制之后，将第 1 位是 1 的所有数字放进 A 方框，将第 2 位是 2 的所有数字放进 B 方框……根据玩家指出这个数字出现的框的编号，得到这个数字的二进制表示，从而迅速算出这个数字是多少。如果不点破，不明就里的游戏参与者是很难发现的。

比如你告诉我一个数在 A、B、C、E 四个方框中出现，那我很快就可以告诉你是 23，这个数是怎么来的？

其实就是 2^0+2^1+2^2+2^4=1+2+4+16=23。

当然，你也可选个简单点，在 B、C 方框内都出现过的。

计算出来的结果就是 2+8=10。

下面我们试着通过编写程序来实现这个游戏，给出玩家所选中数字的所在框的个数 *N* 和这 *N* 个框对应的编号，然后计算出这个数字（框的编号从 1 到 *N*，对应二进制数从右往左数的位数）。

十进制数字与其所对应的二进制数相互转换的过程，解释起来有些麻烦，似乎很多人对逆向转换过程的领悟力远逊于正向的过程。由此，将二进制数与转换成十进制数的过程设计成了一个小游戏“数字心灵感应”，通过在游戏的体验和创意设计的过程中理解二进制数字编码，就去领会计算思维的方法在解决问题中所发挥的独特作用。

对数据执行某种操作，并且对执行操作后得到的结果数据继续执行该操作的方法，就是一种迭代。等到该任务完成后，学生可以更直观地体会到一个机械性的重复操作过程是如何解决数学问题的。

1.2.1 编程前准备

1. 二进制基础

二进制是计算技术中广泛采用的一种数制。二进制数据是用 0 和 1 两个数码来表示的数。它的基数为 2，进位规则是“逢二进一”，借位规则是“借一当二”。

一个 5 位的二进制数表示的最大数是 2^5–1=31，所以

0~31 内的任意一个整数都可以用不超过 5 位的二进制数表示。比如上面那个例子，23 对应的二进制就是 11011，其计算过程可以用表 1-1 所示。

23 的二进制表示为 10111，即 27=16+4+2+1。

表 1-1 二进制数转为十进制数过程

位号	5	4	3	2	1
位权	16	8	4	2	1
对应的二进制数	1	0	1	1	1

“数字心灵感应”游戏玩家猜测的十进制数字对应的二进制的第 x 位为 1，则在设计游戏时该数出现在第 x 组方框上，否则不出现。

再进一步解释：

1——00001

3——00011

5——00101

7——00111

…

29——11101

31——11111

将二进制的第一位（从右往左数）都是 1 的数放进第一组。

将二进制的第二位（从右往左数）都是 1 的数放进第二组。

…

反过来观察，因为 23 的二进制中，第一、二、三、五位

都是 1，所以 27 就会在第 1、2、3、5 组方框中出现。

和十进制相比，二进制具有如下几个优点。

（1）二进制数中只有两个数码 0 和 1，可用具有两个不同稳定状态的元器件来表示一位数码。例如，电路中某一通路的电流的有无，某一节点电压的高低，晶体管的导通和截止等。

（2）二进制数运算简单，大大简化了计算中运算部件的结构。

（3）二进制天然兼容逻辑运算。

同时，由于二进制在日常计数使用上位数往往很长，具有读写不方便的缺点。

2. 导入 Tkinter 库

Tkinter 是 Python 的标准 GUI 库，是非常流行的 Python GUI 工具。Python 使用 Tkinter 可以快速创建 GUI 应用程序。由于 Tkinter 已内置到 Python 的安装包中，只要安装好 Python 之后就能导入 Tkinter 库，而且 IDLE 也是用 Tkinter 编写而成，对于简单的图形界面 Tkinter 能应对自如。

Tkinter 创建顶层窗口可以通过以下步骤。

- 引入 Python 的 Tkinter 模块。
- 创建应用程序的主窗口。
- 在窗口内添加小工具（如标签、按钮、帧等）。
- 呼叫主事件循环，以便捕获在用户的计算机屏幕上的动作。

1）创建一个窗口

```
# 在代码里导入库，起一个别名 tk，以后代码里就用 tk 这
```

```
# 个别名
import tkinter as tk
    # 这个库里面有 Tk() 方法，它的作用是创建一个窗口
root = tk.Tk()
    # 加上这一句，就可以看见窗口了，如图 1-2 所示
root.mainloop()
```

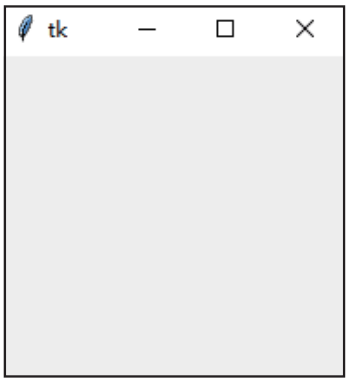


图 1-2　Tkinter 创建一个窗口

2）窗口设置及添加单击事件

```
import tkinter as tk
from tkinter import messagebox
root = tk.Tk()  # 创建窗口
root.title('演示窗口')
root.geometry("300x100+630+80")  # 长 × 宽 +x*y

btn1 = tk.Button(root)  # 创建按钮，并且将按钮放到窗口里面
btn1["text"] = "单击"  # 给按钮起一个名称
btn1.pack()  # 按钮布局
```

```
def test(e):
    "' 创建弹窗 '"
    messagebox.showinfo(" 窗口名称 "," 单击成功 ")


btn1.bind("<Button-1>", test)
                    # 将按钮和方法进行绑定，也就是创建了一个事件
root.mainloop()   # 让窗口一直显示，循环
```

运行效果如图 1-3 所示。

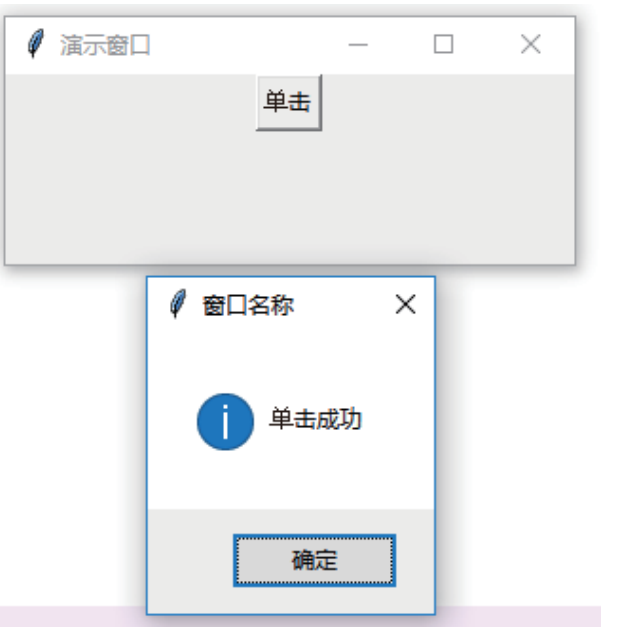


图 1-3　Tkinter 窗口设置及添加单击事件

Tkinter 有各种不同的控件，如按钮、画布、复选框、列表框等，如表 1-2 所示。

表 1-2 Tkinter 控件

编号	控件	描 述
1	Button	按钮：用于增加各种按钮
2	Canvas	画布：用于在窗口上绘制图形，显示图形元素，如线条或文本
3	Checkbutton	复选框：用于显示多选框
4	Entry	输入控件：显示单行文本域。它一般用于接收用户值
5	Frame	框架：在屏幕上显示一个矩形区域，多用来作为容器，其他控件可以加入进来
6	Label	标签：可以显示文本和位图
7	ListBox	列表框：显示一个字符串列表给用户
8	Menubutton	菜单按钮：显示给用户的菜单项目
9	Menu	菜单：显示菜单栏、下拉菜单和弹出菜单
10	Message	消息：用于显示消息给用户，与 Label 比较类似
11	Radiobutton	单选按钮：不同于 Checkbutton，它向用户提供各种选项，并且用户可以只选择其中一个选项
12	Scale	范围控件：显示一个数值刻度，限定数字区间
13	Scrollbar	滚动条：当内容超过可视化区域时使用，用户可以滚动窗口
14	Text	文本：不同于输入，它提供了一个多行文本域给用户，使得用户能够写入文本和编辑文本
15	Toplevel	容器：被用于创建独立窗口，提供一个单独的对话框
16	Spinbox	输入：与 Entry 类似，但是可以指定输入范围值

续表

编号	控件	描　　述
17	PanedWindow	窗口布局管理：就像一个容器，包含水平或垂直控件的窗格
18	LabelFrame	容器控件：常用于复杂的窗口布局
19	MessageBox	消息框：用来显示消息框的桌面应用程序

1.2.2　算法设计

我们可以设置两个角色：

“感应者”为操作游戏的“上帝视角”；

“被感应者”为玩家。

游戏过程：让“被感应者”在十进制的 0~31 内任选一个数字，“感应者”分别告知“被感应者”是否在以下各组中。

第一组：1，3，5，7，9，11，13，15，17，19，21，23，25，27，29，31

第二组：2，3，6，7，10，11，14，15，18，19，22，23，26，27，30，31

第三组：4，5，6，7，12，13，14，15，20，21，22，23，28，29，30，31

第四组：8，9，10，11，12，13，14，15，24，25，26，27，28，29，30，31

第五组：16，17，18，19，20，21，22，23，24，25，26，27，28，29，30，31

感应者往往能在较短时间内猜出这一个数字是什么。

例如，出现“被感应者”选中数字的框对应组号分别是 1，2，4，5，“感应者”就能快速知道猜测的数字是 27。

流程图如图 1-4 所示。

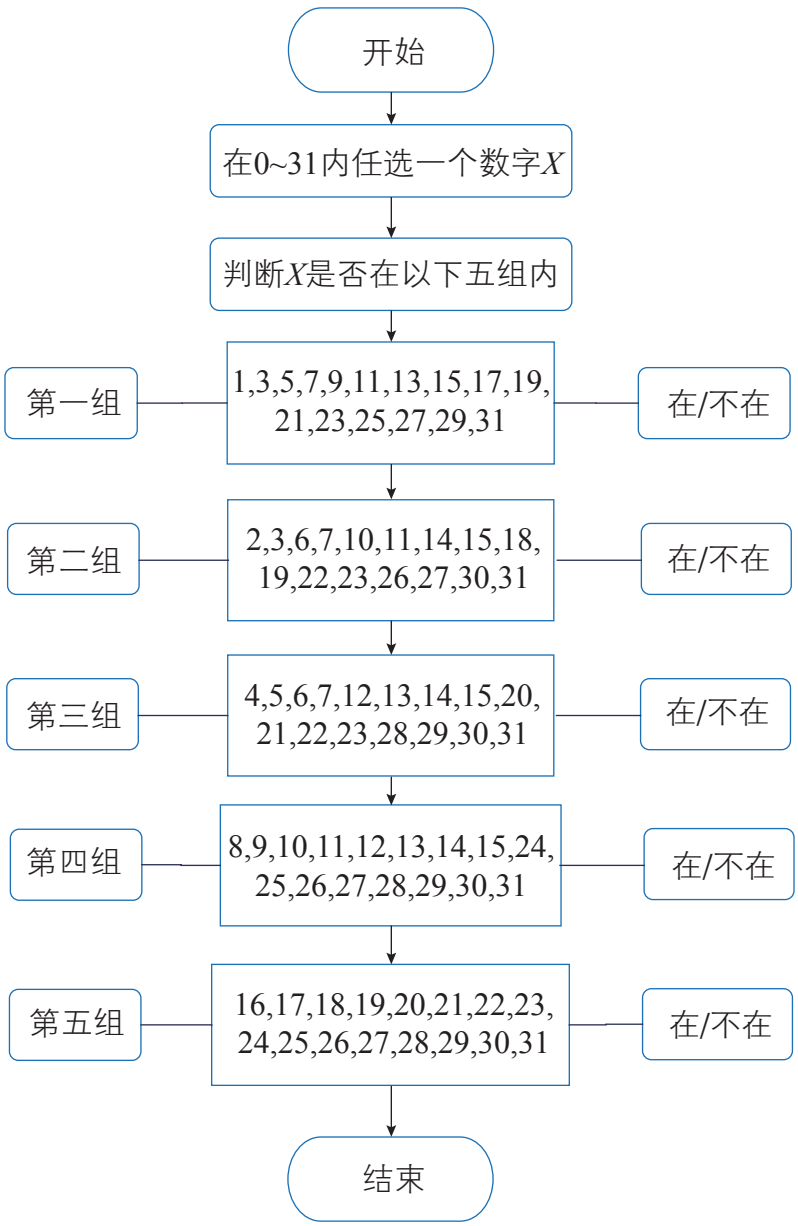


图 1-4 “数字心灵感应小游戏” 流程图

1.3 编写程序及运行

1.3.1 程序代码

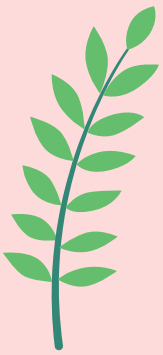
```
import tkinter as tk  # 可视化模块
import tkinter.font as tkFont  # 引入字体模块
import random  # 随机数模块

####################
# 计算部分
####################
all_set=[]  # 存放二进制同位为 1 的数字（元素为同一类的数字列表）
judge=[]  # 存放用户所选数字各位为 1 还是为 0 的列表

# 对各位是否为 1 进行分类，某一位为 1 的放在一起
def classify(number, digit):
    array = []
    for item in number:
    # 将整数转换为字符形式二进制数，并将格式定为 6 位二进制数，检查对应位数是否为 1
        # 十进制数转二进制数
        bin = '{:b}'.format(item)
```

```
            # 转换为字符串形式，并自动在缺位前补零，暂定 6 位
            bin = str('{:0>6}'.format(bin))
            # 判定指定二进制数某一位是否为 1，是则加入 array 成为一类
            if bin.find('1', digit - 1, digit) != -1:
                        array.append(int(bin, 2))
    return array

# 生成可选择数字列表
def generate_number():
    # 容纳全部可选择数字的列表
    numbers=[]
    i = 0
    # 生成 30 个 1~63 的不重复的随机数（二进制最多 6 位）
    while i < 30:
        element = random.randint(1, 63)
        if element not in numbers:# 防止重复
            numbers.append(element)
            i+=1
    # 将随机数从小到大排序
    numbers.sort()
    i = 1
    # 用来将以上数字显示在 GUI 上的字符串
    text=""
    for item in numbers:
        # 拼接各数字
```



```
            text+=str(item)+""
            if i%10==0:
                # 每十个数字换一行
                text+="\n"
            i+=1
        # 赋值给图形界面可更新变量 (choice_number,stringvar 类型 )
        choice_number.set(text)
        # 对以上数字分类
        allset = []
        # 生成 6 个对应集合
        for j in range(1, 7):
            # 对全部数字分类
            allset.append(classify(numbers, j))
        # 复制已分类的数字列表给全局变量 all_set
        all_set.extend(allset)

# 将用户输入进行拼接，并将字符形式的二进制数字转换为整数输出
def transfer():
    result = ""
    for item in judge:
        # 将结果数字二进制各位拼接为字符串
        result += item
    # 将二进制数字字符串转换为十进制数字
    result = int(result, 2)
    # 转换为最终显示在图形界面上的结果字符串
```

```
    result=" 你选择的数字为 "+str(result)+", 我猜对了吧 "
    # 赋值给图形界面可更新变量（result_number,stringvar 类型）
    result_number.set(result)

# 单击 " 在 " 按钮后的处理
def confirm():
    # 单击 " 在 " 按钮代表该位为 1，向列表中添加 1
    judge.append("1")
    # 全部位数确认完毕后的处理
    if len(judge)==6:
        # 列表结果转换为二进制数值再转换为十进制数
        transfer()
        # 重置
        judge.clear()
        # 转结果页
        page4_transfer()
    else:
        # 显示各分类数字（剩余）
        display_classified_number(len(judge))

# 单击 " 不在 " 按钮后的处理
def deny():
    # 单击 " 不在 " 按钮代表该位不是 1，即 0，向列表中添加 0
    judge.append("0")
    # 作用同 confirm 函数
```

```
    if len(judge)==6:
        transfer()
        judge.clear()
        page4_transfer()
    display_classified_number(len(judge))

# 用于显示已分类好的数字列表
def display_classified_number(index):
    classified_number=""
    for number in all_set[index]:
        # 将列表中数字拼接为字符串
        classified_number+=str(number)+""
    # 赋值给图形界面可更新变量(classified_numbers,stringvar类型)
    classified_numbers.set(classified_number)

#############
#GUI部分
#############
# 第一页转到第二页

def page2_transfer():
    # 隐藏"开始"按钮
    start_button.place_forget()
    # 调出选择提示页面
    choose_frame.place(x=190,y=170)
```

```
    # 调出待选择数字界面
    choice_frame.place(x=160,y=100)
    # 生成数字
    generate_number()

# 第二页转到第三页
def page3_transfer():
    # 隐藏选择提示界面
    choose_frame.place_forget()
    # 隐藏待选择数字界面
    choice_frame.place_forget()
    # 调出猜数提示界面
    guess_frame.place(x=170,y=125)
    # 调出各分类数字界面
    number_frame.place(x=135,y=80)
    # 显示各分类数字（第一个）
    display_classified_number(0)

# 第三页转到第四页
def page4_transfer():
    # 隐藏猜数提示界面
    guess_frame.place_forget()
    # 隐藏各分类数字界面
    number_frame.place_forget()
    # 调出猜数结果页面
```

```
    result_frame.place(x=170,y=125)

# 返回初始界面，重新开始新一轮猜数
def restart():
    # 隐藏猜数结果页面
    result_frame.place_forget()
    # 调出开始界面
    start_button.place(x=210,y=120)
    # 重置生成数组
    all_set.clear()

if__name__ == '__main__':
    # 创建主窗口对象
    root=tk.Tk()
    # 调整窗口大小，用字符串 'axb' 形式传递参数
    root.geometry('500x300')
    # 设置窗口标题
    root.title("心灵感应程序")

    choice_number=tk.StringVar()
    result_number=tk.StringVar()
    classified_numbers=tk.StringVar()

    # 开始界面
    # 指定字体名称、大小、样式
```

```
    ft1 = tkFont.Font(family='华 文 新 魏', size=20,
weight=tkFont.NORMAL)
    ft2 = tkFont.Font(family='华 文 仿 宋', size=13,
weight=tkFont.NORMAL)
    title_label=tk.Label(root,text="心灵感应游戏",font=ft1)
    title_label.pack()# 显示 label，须含有此语句
    start_button=tk.Button(root,text="开始",width=8,height
=2,command=page2_transfer,font=ft2)
    start_button.place(x=210,y=120)
    quit_button=tk.Button(root,text="退出",command=root.de
stroy,width=6,height=2,font=ft2)
    quit_button.pack(side="bottom")

    # 选择数字界面
    choose_frame=tk.Frame(root)
    choose_hint=tk.Label(choose_frame,text="请在上方选择一个
数字\n我将猜出你选择的数字")
    choose_hint.pack(side="top")
    confirm_button=tk.Button(choose_frame,text="下 一 步",
command=page3_transfer)
    confirm_button.pack(side="bottom")
    choice_frame=tk.Frame(root)
    choice=tk.Label(choice_frame,textvariable=choice_
number)
    choice.pack()
```

```
    # 猜数过程界面
    guess_frame=tk.Frame(root)
    label=tk.Label(guess_frame,text="你所选择数字是否在以上数组中？")
    yes_button=tk.Button(guess_frame,text="在",command=confirm)
     no_button=tk.Button(guess_frame,text="不在",command=deny)
    label.pack()
    yes_button.pack(side="left")
    no_button.pack(side="right")

    # 显示数字页面
    number_frame=tk.Frame(root)
    number_label=tk.Label(number_frame,textvariable=classified_numbers)
    number_label.pack()

    # 结果页面
    result_frame=tk.Frame(root)
    result_text=tk.Label(result_frame,textvariable=result_number)
    result_text.pack()
    restart_button=tk.Button(result_frame,text="重新开始",
```

```
command=restart)
    restart_button.pack()


    # 开启窗口主循环
    root.mainloop()
```

1.3.2 运行程序

程序运行后，首先会弹出如图 1-5 所示界面，为了增加游戏趣味性，界面中每次会随机生成 30 个 1~63 的不重复的随机数（二进制最多 6 位）。

然后，游戏参与人员“被感应者”可以在这 30 个随机数里面任选一个数，并记住后选择“下一步”。

接着，界面上出现若干个数字，游戏参与人员“被感应者”回答开始选择的那个数字是否在该界面中出现，出现选择“在”，否则选择“不在”，如图 1-6~ 图 1-11 所示选择 6 次。

最后“感应者”根据游戏参与人员“被感应者”的 6 次选择就能够轻松感应（计算）出他心里想的那个数字。

例如，6 次选择分别是“在 - 不在 - 不在 - 在 - 不在 - 不在”，对应的二进制数即为 100100，十进制数为 36，如图 1-12 所示。

图 1-5　程序运行截图 1

图 1-6　程序运行截图 2

图 1-7　程序运行截图 3

图 1-8　程序运行截图 4

图 1-9　程序运行截图 5

图 1-10　程序运行截图 6

图 1-11　程序运行截图 7

图 1-12　程序运行截图 8

拓 展 训 练

问题 1：

一个黑盒子中有一个多项式，次数未知，只知道该多项式所有的系数都是非负整数。每次输入 x，黑盒子会输出 $f(x)$。请问有什么办法可以快速确定这个多项式？

参考答案：

第一次，输入 1，得到整个多项式的所有系数之和 S。

第二次，输入 S+1，输出 M。把 M 转换到（S+1）进制，每一个数位上的数就对应了原多项式的系数。

解释：一个 k 进制的数就是一个多项式把 k 代进去的结果，这个答案就是反其道而行之，通过 k 进制的各位数字来还原多项式。

问题 2：

如果用二进制，大家思考下两只手最多能表示多少个数？为什么呢？

参考答案：

1024 个数。

每根手指代表一个数，即：1-1、2-2、3-4、4-8、5-16、6-32、7-64、8-128、9-256、10-512，所有数加在一起就是 1023。所以两只手最多能代表的数是 0~1023，也就是 1024 个数。

小　结

本章主要讲到了二进制的一个应用，进制是进位记数制，是人为定义的带进位的记数方法。对于任何一种 X 进制，就表示每一位上的数运算时都是逢 X 进 1 位。

例如，“数字心灵感应”这个小游戏用到的二进制就是逢 2 进 1，十进制是逢 10 进 1，还有十六进制（它由 16 个数码：

数字 0~9 加上字母 A~F，A~F 分别表示十进制数 10~15）是逢 16 进 1，八进制就是逢 8 进 1，以此类推，X 进制就是逢 X 进 1。

对于任何一个数，我们可以用不同的进位制来表示。

例如，十进数 27(10)，可以用二进制表示为 11011(2)，也可以用四进制表示为 123(4)，也可以用八进制表示为 33(8)、用十六进制表示为 1B(16)，它们所代表的数值都是一样的。