实验3

练习1

题目

求所有满足下面条件的四位数：它的各位数字的四次方之和恰好等于自己。

解析

使用暴力搜索编写，根据代码示例对代码效率进行了一些改进。

代码

```
public class Exercise_1 {
    public static void main(String[] args) {
        int[] c = new int[10];
        for (int a = 0; a < 10; a++) {
            c[a] = a * a * a * a;
        }
        int n = 999;
        for (int i = 1; i <= 9; i++) {
            int ic = c[i];
            for (int j = 0; j <= 9; j++) {
                int jc = c[j];
                for (int k = 0; k <= 9; k++) {
                    int kc = c[k];
                    for (int p = 0; p < 10; p++) {
                        n++;
                        if (ic + jc + kc + c[p] == n) {
                            System.out.println(n);
                        }
                    }
                }
            }
        }
    }
}
```

输入

```
无
```

输出

```
1634
8208
9474
```

练习2

题目

求所有满足下面条件的三位数：它能被 11 整除，且所得的商恰好等于它的各位数字的平方和。

解析

使用了稍复杂的算法，详见代码注释。

主要原理是使用hundreds、tens、ones三个变量分别表示个位、十位、百位。而商则用n单独表示。在程序中避免了除法的出现，以提高代码效率。

代码

```
public class Exercise_2 {
    public static void main(String[] args) {
        int[] c = new int[10]; // 用来保存0~9的平方
        for (int a = 0; a < 10; a++) {
            c[a] = a * a;
        }
        int n = 9; // n为三位数除以11所得的商
        int hundreds; // 百位
        int tens = -1; // 十位
        int ones = 9; // 个位
        // 初始值为109
        for (hundreds = 1; hundreds <= 9; hundreds++) { // 百位从1到9递增
            int hundredsC = c[hundreds]; // 百位的平方
            for (int i = 0; i < 9; i++) { // 由于是三位数，每一次百位循环都只能加8次11
（109+8*11=197；902+8*11=990）
                tens++; // 每次十位加1
                ones++; // 每次个位加1
                if (ones >= 10) { // 若个位>=10则进位
                    ones %= 10;
                    tens++;
                }
                if (tens >= 10) { // 若十位>=10则进位
                    tens %= 10;
                }
                n++; // 商加1
                if (hundredsC + c[tens] + c[ones] == n) { // 若商等于平方和，则打印结
果
                    System.out.println(n * 11);
                }
            }
        }
    }
}
```

虽然代码效率非常高，但为这么简单的问题投入如此大的精力似乎并不值得。在实际问题中肯定是不会这么做的。

输入

```
无
```

输出

```
550
803
```

练习3

题目

设 a,b 都是不超过 100 的正整数。
在整数范围内，设(a^2+b^2)除以(a+b)所得的商为 q，
余数为 r。求满足 $q^2+r=2008$ 的所有正整数对(a,b)。

解析

采用暴力搜索，依照代码示例进行了部分优化。

代码

```
public class Exercise_3 {
    public static void main(String[] args) {
        int[] c = new int[101];
        for (int a = 0; a <= 100; a++) {
            c[a] = a * a;
        }
        for (int a = 1; a <= 100; a++) {
            int ac = c[a];
            for (int b = a; b <= 100; b++) {
                int i = ac + c[b];
                int j = a + b;
                int q = i / j;
                int r = i % j;
                if (q * q + r == 2008) {
                    System.out.println(String.format("(%d, %d)", a, b));
                }
            }
        }
    }
}
```

输入

```
无
```

输出

```
(26, 54)
(38, 50)
```

练习4

题目

已知一个递增序列，元素两两不等，它们满足下面的条件：
（1）数 1 在序列中。
（2）若数 x 在序列中，则 2x, 3x, 5x 也在序列中。
（3）除此之外，序列中无其他数。求该序列开头的 100 个元素。

解析

将元素部分列出后，找到规律：序列中的元素都由2、3、5的n次方（n为自然数）组成。

因此程序只需找出前100个符合条件的值即可。

代码

```
public class Exercise_4 {
    public static void main(String[] args) {
        int[] numbers = new int[100];
        numbers[0] = 1;
        int n = 1;
        int count = 1;
        while (count < 100) {
            n++;
            int t = n;
            while (t % 2 == 0) {
                t /= 2;
            }
            while (t % 3 == 0) {
                t /= 3;
            }
            while (t % 5 == 0) {
                t /= 5;
            }
            if (t == 1) {
                numbers[count] = n;
                count++;
            }
        }
        for (int i: numbers) {
            System.out.print(i + " ");
        }
    }
}
```

输入

```
无
```

输出

```
1 2 3 4 5 6 8 9 10 12 15 16 18 20 24 25 27 30 32 36 40 45 48 50 54 60 64 72 75 80
81 90 96 100 108 120 125 128 135 144 150 160 162 180 192 200 216 225 240 243 250
256 270 288 300 320 324 360 375 384 400 405 432 450 480 486 500 512 540 576 600
625 640 648 675 720 729 750 768 800 810 864 900 960 972 1000 1024 1080 1125 1152
1200 1215 1250 1280 1296 1350 1440 1458 1500 1536
```

练习5

题目

Finding the gcd
Given *n* positive integers between 1 and 200000 (1 ≤ n ≤ 100),
you are required to find the greatest common divisor of the *n* integers.
For example, if *n* = 3, and the integers are 18, 63, 36,
then the greatest common divisor is 9.

解析

使用辗转相除法可以简单地解决问题。

仅需多次运用辗转相除法求出两个数之间的最大公约数，即可求出所有数的最大公约数。

代码

```java
import java.util.Scanner;

public class Exercise_5 {
    public static int gcd(int m, int n) {
        if (m < n) {
            int k = m;
            m = n;
            n = k;
        }
        if (m % n != 0) {
            m %= n;
            return gcd(m, n);
        }
        return n;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] nums = new int[n];
        for (int i = 0; i < n; i++) {
```

```
            nums[i] = sc.nextInt();
        }
        sc.close();
        int divisor = nums[0];
        for (int i = 1; i < n; i++) {
            divisor = gcd(divisor, nums[i]);
        }
        System.out.println(divisor);
    }
}
```

---

*练习6

题目

Number Lists
Given P integers 1, 2, 3, ..., P, you can construct a list
which contains L integers chosen from the P integers,
but the list can not have K or more than K consecutive 1's.
(1 ≤ P < 10, 1 < L < 31, 1 < K < L + 1)
For example, when P = 2, L = 3, and K = 2, the lists can be:
121
122
212
221
222
There are 5 lists.
In the case of P = 3, L = 3, and K = 3, the lists can be:
112 211 311
113 212 312
121 213 313
122 221 321
123 222 322
131 223 323
132 231 331
133 232 332
233 333
There are 26 lists.
Given three integers P, L, and K, you are required to
calculate the total number of allowable lists as described above.

解析

根据题意，可直接推导出结果N与P、L、K的关系：

$$N = P^L - (P^{L-K+1} - (P-1)^{L-K+1})$$

数学推导仅需用到一些基本的组合数学知识，在此省略。

代码

```
import java.util.Scanner;

public class Exercise_6 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int P = sc.nextInt();
        int L = sc.nextInt();
        int K = sc.nextInt();
        sc.close();
        int result = 0;
        result = (int) (Math.pow(P, L - K + 1) - Math.pow(P - 1, L - K + 1));
        result = (int) (Math.pow(P, L) - result);
        System.out.println(result);
    }
}
```

输入1

```
2 3 2
```

输出1

```
5
```

输入2

```
3 3 3
```

输出2

```
26
```

心得体会

1. 多层循环中很多值进行了重复计算，可以稍加修改使计算仅进行一次，用空间换时间。
2. 找到问题的规律之后解答会变得相当简单（例如发现递推式或公式）。