实验4

练习1

题目

打印下面的图案：

```
      **
    **  **
  ***    ***
****      ****
  ***    ***
    **  **
      **
```

解析

循环的简单应用，没什么好说的。

代码

```
public class Exercise1 {

    public static void main(String[] args) {
        int n = 4;
        int[] c = new int[n * 2 - 1];
        for (int i = 0; i < n; i++) {
            c[i] = i + 1;
        }
        for (int i = n; i < c.length; i++) {
            c[i] = c[i - 1] - 1;
        }
        for (int k = 0; k < c.length; k++) {
            int i = c[k];
            for (int j = 0; j < (n - i) * 2; j++) {
                System.out.print(" ");
            }
            for (int j = 0; j < i; j++) {
                System.out.print("*");
            }
            for (int j = 0; j < (i - 1) * 2; j++) {
                System.out.print(" ");
```

```
                }
                for (int j = 0; j < i; j++) {
                    System.out.print("*");
                }
                System.out.println();
            }
        }

    }
```

输入

```
无
```

输出

```
      **
    **  **
  ***    ***
****      ****
  ***    ***
    **  **
      **
```

实验2

题目

某人岁数的二次方是个四位数，三次方是个六位数，该四位数和六位数恰好用遍 0 至 9 共 10 个数字。求该人岁数。

解析

暴力搜索。根据题意可知岁数在47至99之间。

对每一种情况，使用字符串将二次方和三次方的数字各位分离，记录到字符串数组a和b中，然后将a和b合并成数组c，最后使用Java内置的HashMap类型进行去重，若去重后仍有10个元素，则打印结果。

代码

```
import java.util.Arrays;
import java.util.HashSet;
```

```
public class Exercise2 {

    public static void main(String[] args) {
        for (int i = 47; i < 100; i++) {
            String[] a = String.valueOf(i * i).split("");
            String[] b = String.valueOf(i * i * i).split("");
            String[] c = Arrays.copyOf(a, 10);
            for (int j = 4; j < 10; j++) {
                c[j] = b[j - 4];
            }
            HashSet<String> s = new HashSet<>(Arrays.asList(c));
            if (s.size() >= 10) {
                System.out.println(i);
                break;
            }
        }
    }

}
```

输入

```
无
```

输出

```
69
```

练习3

题目

输入 20 个整数，按最小值、最大值、次最小值、次最大值……的次序输出。

解析

先对输入数组nums进行排序（调用了Java内置的sort函数，省略了快速排序的代码）。

然后依次打印第1个数字和第20个数字，第2个和第19个数字……依此类推。

使用优先队列似乎是更好的想法，但考虑到会使得代码过度复杂，并不值得。

代码

```
import java.util.Arrays;
import java.util.Scanner;

public class Exercise3 {

    public static void main(String[] args) {
        int n = 20;
        int[] nums = new int[n];
        Scanner in = new Scanner(System.in);
        for (int i = 0; i < n; i++) {
            nums[i] = in.nextInt();
        }
        in.close();
        Arrays.sort(nums);
        for (int i = 0; i < n / 2; i++) {
            String s = nums[i] + " " + nums[n - i - 1] + " ";
            System.out.print(s);
        }
    }
}
```

输入

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

输出

```
1 20 2 19 3 18 4 17 5 16 6 15 7 14 8 13 9 12 10 11
```

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+1, 3x+1 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

和实验3中的练习4很相似，但实际上完全不同。本题无法找到可以直接导出元素的规律，因此按照题意暴力查找前100个元素。

算法中会涉及到多次元素搬移，因此效率不高。

代码

```
public class Exercise4 {

    public static void main(String[] args) {
        int n = 100;
        int[] nums = new int[n];
        nums[0] = 1;
        for (int i = 0; i < n; i++) {
            int num = nums[i];
            int a = num * 2 + 1;
            int b = num * 3 + 1;
            for (int j = i; j < n; j++) {
                if (a < nums[j] || nums[j] == 0) {
                    for (int k = n - 1; k > j; k--) {
                        nums[k] = nums[k - 1];
                    }
                    nums[j] = a;
                    break;
                }
            }
            for (int j = i; j < n; j++) {
                if (b < nums[j] || nums[j] == 0) {
                    for (int k = n - 1; k > j; k--) {
                        nums[k] = nums[k - 1];
                    }
                    nums[j] = b;
                    break;
                }
            }
        }
        for (int i = 0; i < n; i++) {
            System.out.print(nums[i]);
            System.out.print(" ");
        }
    }

}
```

输入

```
无
```

输出

```
1 3 4 7 9 10 13 15 19 21 22 27 28 31 31 39 40 43 45 46 55 57 58 63 63 64 67 79 81
82 85 87 91 93 94 94 111 115 117 118 121 127 127 129 130 135 136 139 159 163 165
166 171 172 175 175 183 187 189 189 190 190 193 202 223 231 235 237 238 243 244 247
255 255 256 259 261 262 271 273 274 279 280 283 283 319 327 331 333 334 343 345 346
351 351 352 355 364 367 375
```

心得体会

1. 调用语言本身提供的数据结构可以更方便地解决部分问题。
2. 在解决问题之前进行必要的范围限定可以极大地提高程序效率。