

Java Genesis Chapter 6

Challenge Problems

Problem 1: finding permutations

(a) listing all permutations

Consider the following Java code:

```
import genesis.*;
public class ListingAllPermutations {

    public static void main (String [ ] args) {
        String s = DialogBox.request("string to be permuted:");
        Transcript.println(listAll(s));
    }
}
```

Code the method `listAll` which when passed a string `s` of distinct characters (none of which is the comma character ',') returns a string in which all the permutations of the characters in the string `s` are listed, each separated by a comma. For example, if the string `abcd` is supplied when requested by the dialog box, the following output will appear in the Transcript window:

```
abcd, abdc, acbd, acdb, adbc, adcb, bacd, badc, bcad, bcda, bdac, bdca,
cabd, cadb, cbad, cbda, cdab, cdba, dabc, dacb, dbac, dbca, dcab, dcba
```

(b) ordered selections of a given size

As a variation on the above, consider the following Java code:

```
import genesis.*;
public class OrderedSelections {

    public static void main (String [ ] args) {
        String s = DialogBox.request("string of characters:");
        int num = DialogBox.requestInt("number to be selected:");
        Transcript.println(listAll(s, num));
    }
}
```

Code the method `listAll` which when passed a string `s` of distinct characters (none of which is the comma character ',') and a positive integer `num` whose value does not exceed the length of the string `s`, returns a string in which all the ordered selection of `num` characters from the string `s` are listed, each separated by a comma. For example, if the string `abcde` is supplied when requested by the first dialog box, and the integer 3 is supplied when requested by the second dialog box, the following output will appear in the Transcript window:

```
abc,abd,abe,acb,acd,ace,adb,adc,ade,aeb,aec,aed,bac,bad,bae,
bca,bcd,bce,bda,bdc,bde,bea,bec,bed,cab,cad,cae,cba,cbd,cbe,
cda,cdb,cde,cea,ceb,ced,dab,dac,dae,dba,dbc,dbe,dca,dcb,dce,
dea,deb,dec,eab,eac,ead,eba,ebc,ebd,eca,ecb,ecd,eda,edb,edc
```

Problem 2: preferential voting

In Australian elections, if there are n candidates each voter must assign to each candidate precisely one of the number 1 to n , where 1 indicates first preference, 2 second preference, and so on. Counting consists of one or more stages. If after any stage a candidate receives strictly more than half the of the first preference votes then that candidate is declared the winner; otherwise those candidates (one or more) with the least first preference votes are eliminated and each vote currently assigned to them is re-assigned as a first preference to that voter's next most preferred candidate. If after any stage all the remaining candidates have the same number of first preference votes then these candidates are declared the joint winners (and a new election is called).

Write a program to determine the first preference votes at each stage and the final winning candidate(s), given the voting preferences of each of the voters. For example. suppose there are four candidates (C1, C2, C3 and C4) and 11 voters who voted as follows:

C1	C2	C3	C4
1	2	3	4
1	4	2	3
1	3	4	2
1	2	4	3
4	1	2	3
3	1	4	2
2	1	3	4
3	1	4	2
4	2	1	3
4	3	1	2
3	4	2	1

Then when supplied with this data the program should output the following to the Transcript window:

```
stage 1    4  4  2  1
stage 2    4  4  3  0
stage 3    5  6  0  0
Winner is candidate 2
```

Note: you may find it useful to make use of 2-dimensional arrays when constructing your code. For example, the intentions of each voter in the example above could be recorded as the following 2-dimensional array:

```
int [ ][ ] votes = {{1,2,3,4},{1,4,2,3},{1,3,4,2},{1,2,4,3},
                    {4,1,2,3},{3,1,4,2},{2,1,3,4},{3,1,4,2},
                    {4,2,1,3},{4,3,1,2},{3,4,2,1}};
```

This problem is similar to one set as part of the 1999 ACM South Pacific Programming Competition.

Problem 3: exact integer arithmetic

Exact integer arithmetic can be done by using Java's `BigInteger` class. The aim here, however, is to do exact integer arithmetic using only Java's `int` type. The trick is to record the usual decimal representation of a large integer as an array of digits, i.e. as an array of type `int` where each element of the array is some integer in the range 0 to 9 inclusive.

(a) exact factorials

Write a program to compute the exact factorial of a positive integer. When the program is executed a dialog box opens requesting an integer (up to, say, 5000) and when the integer is supplied, the program outputs to the Transcript window the exact factorial of that integer.

(b) exact integer multiplication

Write a program to multiply exactly any two integers of any size. When the program executes a dialog box opens requesting the first integer which the user inputs as a string of digits of arbitrary length, then another dialog box opens requesting the second integer and again the user supplies it as a string of digits of arbitrary length. The program then prints in the Transcript window both of the supplied integers followed by the exact product of these two integers.

Problem 4: integers with distinct digits

We are interested in integers with the following 3 properties:

- the integer's decimal representation contains exactly five digits;
- each of these five digits is different, i.e. no digit is repeated;
- when multiplied by 2 another five digit integer is obtained consisting precisely of the five digits not in the original integer.

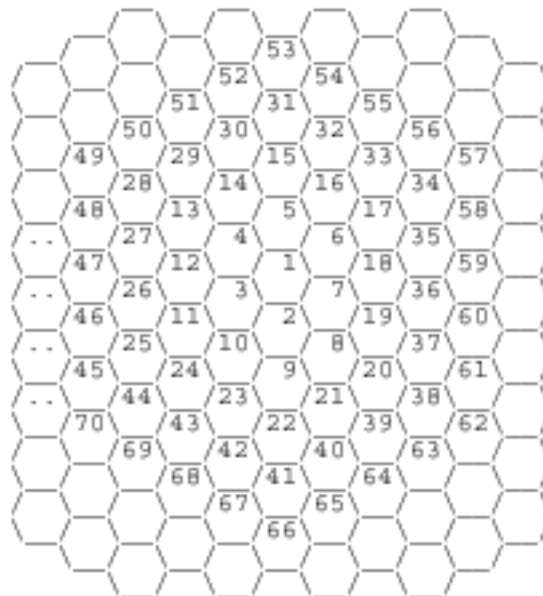
For example, the integer 32079 satisfies the above properties: it consists of five distinct digits and when multiplied by 2 gives 64158 which is a five digit integer containing precisely the remaining digits.

Write a Java program to list in the Transcript window all the integers with the above 3 properties.

Problem 5: distances in the beehive

This problem was set in 1999 as part of the world finals of the 23rd Annual ACM International Collegiate Programming Contest.

The cells of a beehive are numbered by marking an arbitrary cell as number 1, and then labeling the remaining cells in a clockwise fashion as in the following figure:



Write a Java program that when executed request via two dialog boxes two positive integers each less than 10000 denoting two cells in the behave, and then prints in the Transcript window the distance between the two cells, where this distance is defined as the number of cells visited in a shortest path between the cells. For example, if the integers 19 and 30 are supplied, the following should appear in the Transcript window:

The distance between cells 19 and 30 is 5.

As can be checked from the figure above, there are several paths of length 5 from cell 19 to cell 30 (e.g. 19-7-6-5-15-30) but none of smaller length.

Problem 6: sliding sums

Consider the 3 by 3 sliding tile puzzle (see Problem 2 in Chapter 1 of Java Genesis). The aim of this puzzle as presented in Chapter 1 is to slide the tiles until the tiles are all in order, as shown in the following figure:

1	2	3
4	5	6
7	8	

Consider now the problem of sliding the tiles to form configurations with the property that if we take the blank space as representing 0 and each row as representing the 3-digit integer determined by the digits on the tiles, the first two rows sum to give the third row. Two examples of such tile configurations that can be obtained by sliding the tiles are given in the following figure:

1	5	6
	7	8
2	3	4

1	2	5
4	7	8
6		3

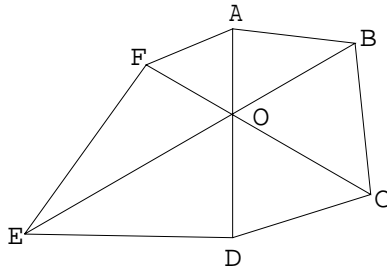
That is, $156 + 78 = 234$ and $125 + 478 = 603$.

Write a Java program to find all such configurations of the tiles, i.e. configurations that arise by sliding the tiles so that the first two rows sum to the third row (where we assign the digit 0 to the blank tile). For each such configuration, print the sum that results in the Transcript window (e.g. $156 + 78 = 234$).

(Hint: there are 84 such configurations in all.)

Problem 7: integer-sided hexagon

The irregular hexagon $ABCDEF$ (see the figure below) is such that the three diagonals AD , BE and CF meet in a common point O . Furthermore, each of the 12 lines AB , BC , CD , DE , EF , FA , AO , BO , CO , DO , EO and FO has an integer less than 50 as its length, the lengths of the six sides that meet at O (i.e. AO , BO , CO , DO , EO and FO) are all distinct, and finally, the six angles at O (i.e. the angles AOB , BOC , COD , DOE , EOF and FOA) are each 60° .



It turns out, there is essentially only one irregular hexagon that satisfies these conditions. Write a Java program that prints in the Transcript window the dimensions of the hexagon, i.e. the lengths of the 12 lines AB , BC , CD , DE , EF , FA , AO , BO , CO , DO , EO and FO .