Assignment overview

It's just past Groundhog Day, the traditional time when Santa's elves return to work after a restful, well-deserved vacation. First on the to-do list: deck the halls of Santa's Workshop with a dazzling display of Christmas lights.

Each light bulb can display three different colours (red, white, green), and the colours along a string are determined by a *pattern*. A pattern has one digit for each bulb, with red=0, white=1, green=2. For example, the following pattern for a string of 8 lights means that the 1st, 3rd, 5th, and 7th lights are red and the rest are green:

```
02020202
```

To animate the lights you can specify a sequence of these patterns. The light string will cycle through the patterns in the list. E.g. here is a sequence of three patterns that will make the colours look like they are marching along the string.

```
01201201
12012012
20120120
```

Your assignment is to write a program that automatically generates lists of patterns like these.

Part 1 (4 marks)

Santa is a pattern-junkie with just a hint of OCD, and he wants to see ALL THE PATTERNS. You need to write a program that will produce all the possible patterns for a given number (N) of lights.

Write a recursive function (i.e. using a decrease-and-conquer algorithm) that will generate a list of all the patterns of a given length. It does not matter what order the patterns appear on the list.

For example, if your function is called with an argument of N=3, the following would be the contents of the ArrayList. These are strings, not integers:

```
[000, 001, 002, 010, 011, 012, 020, 021, 022, 100, 101, 102, 110, 111,
112, 120, 121, 122, 200, 201, 202, 210, 211, 212, 220, 221, 222]
```

Here is the list for N=4:

```
[0000, 0001, 0002, 0010, 0011, 0012, 0020, 0021, 0022, 0100, 0101,
0102, 0110, 0111, 0112, 0120, 0121, 0122, 0200, 0201, 0202, 0210, 0211,
0212, 0220, 0221, 0222, 1000, 1001, 1002, 1010, 1011, 1012, 1020, 1021,
1022, 1100, 1101, 1102, 1110, 1111, 1112, 1120, 1121, 1122, 1200, 1201,
1202, 1210, 1211, 1212, 1220, 1221, 1222, 2000, 2001, 2002, 2010, 2011,
2012, 2020, 2021, 2022, 2100, 2101, 2102, 2110, 2111, 2112, 2120, 2121,
2122, 2200, 2201, 2202, 2210, 2211, 2212, 2220, 2221, 2222]
```

Your function MUST take a single argument (an integer) and MUST return an ArrayList of Strings (containing all the patterns).

Your function MUST use a decrease-and-conquer approach to generate the strings. Recall the general idea of a decrease-and-conquer algorithm to solve a problem of input size N:

1. Recursively solve the problem for an input size of N-1.
2. Adjust or extend the results for N-1 to obtain the solution for input size N.

Part 2 (4 marks)

Your program is a HUGE hit with the elves, but it quickly reveals a manufacturing glitch in the microprocessor that controls the light display. If the controller reads any pattern that contains two of the same digit in a row, it goes into an error condition that requires a "hard" reboot (i.e. somebody has to unplug the whole thing and plug it back in!) This puts a serious dent in the elves' productivity, so you decide to make an alternative list that contains *all the patterns that do not have any double digits*. (Hopefully Santa won't notice!)

There aren't nearly as many of these patterns. Here is the output for N=4:

```
[0101, 0102, 0120, 0121, 0201, 0202, 0210, 0212, 1010, 1012, 1020,
1021, 1201, 1202, 1210, 1212, 2010, 2012, 2020, 2021, 2101, 2102, 2120,
2121]
```

And for N=5:

```
[01010, 01012, 01020, 01021, 01201, 01202, 01210, 01212, 02010, 02012,
02020, 02021, 02101, 02102, 02120, 02121, 10101, 10102, 10120, 10121,
10201, 10202, 10210, 10212, 12010, 12012, 12020, 12021, 12101, 12102,
12120, 12121, 20101, 20102, 20120, 20121, 20201, 20202, 20210, 20212,
21010, 21012, 21020, 21021, 21201, 21202, 21210, 21212]
```

Write another decrease-and-conquer function that produces this list for a given number of lights. As in Part 1 this function MUST take a single argument (an integer) and MUST return an ArrayList of Strings.

Part 3 (3 marks)

Include code in your main() that runs your functions for different sizes of light-strings and produces the following output:

```
Generating all patterns of a given length:
Length 1 produces 3 patterns.
Length 2 produces 9 patterns.
Length 3 produces 27 patterns.
Length 4 produces 81 patterns.
Length 5 produces 243 patterns.
Length 6 produces 729 patterns.
Length 7 produces 2187 patterns.
Length 8 produces 6561 patterns.
Length 9 produces 19683 patterns.
Length 10 produces 59049 patterns.

Generating patterns without double-digits:
Length 1 produces 3 patterns.
Length 2 produces 6 patterns.
Length 3 produces 12 patterns.
Length 4 produces 24 patterns.
Length 5 produces 48 patterns.
Length 6 produces 96 patterns.
Length 7 produces 192 patterns.
Length 8 produces 384 patterns.
Length 9 produces 768 patterns.
Length 10 produces 1536 patterns.
```

Part 4 (4 marks)

You must use good programming style throughout your code, including:

- Header comments with at least the following information:
    o Your name, ID, and set
    o A short description of the program (a sentence is enough)
- A descriptive comment for any significant block of code such as a function or an important loop
- Comments for any other code that is "unusual" or otherwise interesting
- Meaningful variable names for important variables

Submission and marking

Due date: As shown on Learning Hub. Your last submission will be the one that counts. Late assignments will not be graded.

Submit: Your Java source code (file name not important)

Marking: This lab is worth 15 marks.