Parker Bedlan

Mr. Speice

Independent Study and Mentorship, 4A

10 February 2017

Research Assessment #9

Subject: Hidden Conditions and Arrays vs. Lists

MLA Citation:

"HP CodeWars 2007 Problems: LED Decoder" *HPCodeWars.org*. HP, 2007. Web.

Assessment:

When I started my solution for the LED Decoder, it seemed like a very easy program that I could knock out in around twenty minutes. The program was simply to convert numbers that would represent different lines on LED lights to the letters they represent, something very similar to programs I have done before. However, the format of the input was extremely confusing due to it all being on one line without spaces to separate each letter, and it was unclear whether a given number was part of one letter or another. After playing a long game of whack-a-mole with my debugging, I was finally able to have an effective solution put together, but in two hours rather than twenty minutes.

My strategy was naturally to break the input into a character array and take it one character at a time with a for loop. It started simple: if it was a letter, add it to the output, and if it was a number, keep collecting numbers until a smaller number comes up than the last one, marking the beginning of a new letter and matching the previous collection of numbers with the

letter their LED shape corresponded to. The strategy seemed like it would work, but there were more hidden conditions in this program than I have ever witnessed, and I was not expecting a single one of them. I had to make a letter interrupt a series of numbers, as well as the end of the loop end it rather than just smaller numbers, that way no numbers are left out in any situation. I also realized that the program requested that the order is 1234567890, which is in numerical order until it gets to the 0 at the very end, so I had to make an exception for the 0. The 0 was the most painful part of the program, because it acted as a whitespace in situations where it did not fit as a letter, and I had to find a way to effectively do a check for this condition.

On top of all of these exceptions, I learned some tedious details about arrays and lists in Java. The first is that when you compare arrays with the "array1.equals(array2)" method, it checks to see if the arrays are under the same memory space, or the same exact array with a second name. What I wanted, which I realized after a lot of bug testing and google searching, was Array.equals(array1, array2), which compared the contents of the arrays rather than the memory space. I also learned that editing a given list that was entered as a parameter for a method edits the original list as well, unlike arrays. This meant that I would have to manipulate an identical list with a different space in memory if I did not want to make permanent changes, or simply undo any changes made, which was the decision that I made. Using lists and arrays in the same program turned out to be confusing because of these differences, and in the future I plan on staying consistent with one.

Overall, I have learned three lessons from this program. The first is that I should never underestimate the conditions of a program and all of the minute details that will go into making it. I also learned that writing out all of the conditions onto a whiteboard or a separate document is

a much more effective way of handling all of the small requirements than just typing up a program and taking hours to debug it. Finally, I learned that consistency is key when it comes to lists and arrays due to all of their tediously small differences in behavior.

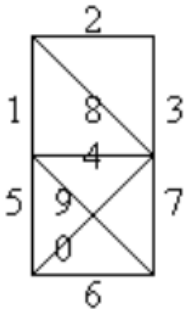
LED Decorator

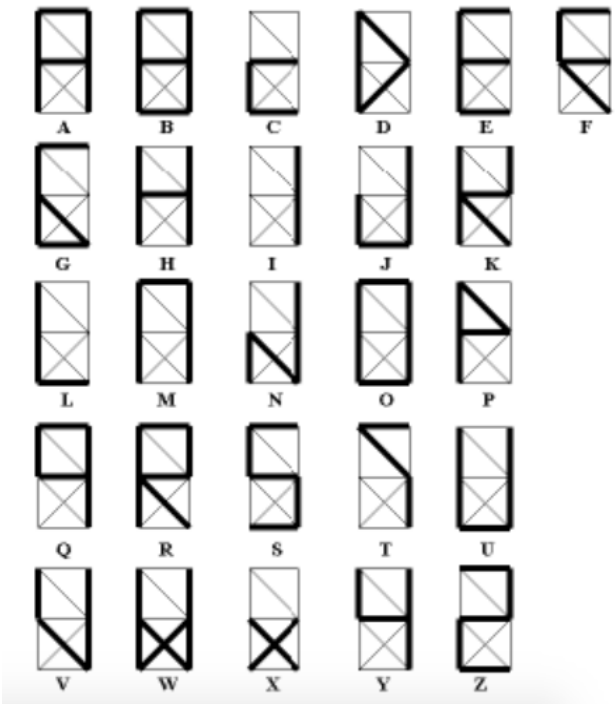
Overview

Some LED systems display characters as a combination of light segments, much like some calculators or gas pumps do. Suppose we have such an LED system, in which each letter of the standard English alphabet is constructed by combining some of ten possible light segments, numbered as shown below: For instance, the letter A is shown using the lines 1,2,3,4,5 and 7. With these few segments, it is not possible, of course, to show all the 26 letters with their natural shapes. A complete list of the letters of our system is shown below. Your task is to translate a combination of numbers (representing light segments) into their respective letters, finally forming a whole word or phrase. Your input is a string of letters and/or numbers. Your output will be a string of letters.

Comment [1]: needs to be converted to 2D array

Comment [2]: This description sounds easy enough.





Input

The input is a file containing a single line representing an input phrase. Each input phrase consists of all uppercase letters, blank spaces and/or digits. In the case of digits, their combination must form valid LED letters. Each letter is encoded as a combination of numbers, ordered in the form 1,2,3,4,5,6,7,8,9,0. A zero (0) that is not part of a valid letter code is interpreted as a blank space. You may assume that no invalid codes are entered, and that the system does not allow ambiguity between two letter codes.

```
HELL1235670WO1234591561580
PROGRAMMING037124670C123567123567156
AND MORE037124903735790278134573712467045612356735792781245612467278
```

Comment [3]: What if they don't? I need a way to handle this exception.

Comment [4]: Not in perfect least-to-greatest order; I'll have to make an exception for 0

Comment [5]: I'll have to make a check for whether the zero is part of a valid code, then decide on whether it is a space or part of a letter.

Comment [6]: This exception is unneeded.

Output

Output the results of the decoding process. The alphabetic letters and blank spaces must not be decoded at all: only the numbers must be converted to the corresponding LED letters.

```
HELLO WORLD
PROGRAMMING IS COOL
AND MORE IF IN THIS CONTEST
```

```
//Code Wars 2007, 13 points
//took about 2 hours

import java.io.*;
import java.util.*;
public class LEDDecorator
{
    static Integer[][] alphaList =
    {
        {1, 2, 3, 4, 5, 7},
        {1, 2, 3, 4, 5, 6, 7},
        {4, 5, 6},
        {1, 5, 8, 0},
        {1, 2, 4, 5, 6},
        {1, 2, 4, 9},
        {1, 2, 5, 6, 9},
        {1, 3, 4, 5, 7},
        {3, 7},
        {3, 5, 6, 7},
        {1, 3, 4, 5, 9},
        {1, 5, 6},
        {1, 2, 3, 5, 7},
        {3, 5, 7, 9},
        {1, 2, 3, 5, 6, 7},
        {1, 4, 5, 8},
        {1, 2, 3, 4, 7},
        {1, 2, 3, 4, 5, 9},
        {1, 2, 4, 6, 7},
        {2, 7, 8},
        {1, 3, 5, 6, 7},
        {1, 3, 7, 9},
        {1, 3, 5, 7, 9, 0},
        {9, 0},
        {1, 3, 4, 7},
        {2, 3, 4, 5, 6}
    };

    public static void main(String args[]) throws F
ileNotFoundException
    {
        Scanner kb = new Scanner(System.in);
```

```
        while(kb.hasNext())
        {
            String input = kb.nextLine();
            char[] chars = input.toCharArray();
            String output = "";
            ArrayList<Integer> nums = new ArrayList
<Integer>();

            for(int i = 0; i < chars.length; i++)
            {
                if(Character.isLetter(chars[i]) ||
Character.isWhitespace(chars[i]))
                {
                    if(nums.size() > 0)
                    {
                        output += getLetter(nums);
                        nums = new ArrayList<Intege
r>();
                    }

                    output += chars[i];
                }
                else
                {
                    int num = Character.getNumericV
alue(chars[i]);

                    if(nums.size() == 0)
                    {
                        if(num == 0)
                        {
                            output += " ";
                        }
                        else
                        {
                            nums.add(num);
                        }
                    }
                    else
                    {
                        if(num < nums.get(nums.size
()-1) && num > 0)
```

```
                        {
                            output += getLetter(num
s);
                            nums = new ArrayList<In
teger>();
                            nums.add(num);
                        }
                        else if(num == 0 && !works(
nums, num))
                        {
                            output += getLetter(num
s);
                            nums = new ArrayList<In
teger>();
                            output += " ";
                        }
                        else
                        {
                            nums.add(num);
                        }
                    }
                }
                if(i == chars.length-1)
                {
                    output += getLetter(nums);
                }

            }

            System.out.println(output);
        }

    }

    static boolean works(ArrayList<Integer> numArra
yList, int num)
    {
        numArrayList.add(num);
        boolean output = Character.isLetter(getLett
er(numArrayList));
        numArrayList.remove(numArrayList.size()-1);
```

```
        return output;
    }

    static char getLetter(ArrayList<Integer> numArr
ayList)
    {
        Integer[] numArray = numArrayList.toArray(n
ew Integer[numArrayList.size()]);

        int alphaNum = -1;

        for(int y = 0; y < 26; y++)
        {
            if(Arrays.equals(alphaList[y], numArray
))
            {
                alphaNum = y;
                break;
            }
        }

        if(alphaNum > -1)
        {
            alphaNum += 65;
            return (char)alphaNum;
        }
        else
        {
            return (char)37;
        }

    }
}
```