COP-3337 Programming II

Programming Assignment 4: Recursion

FIU Knight Foundation School of Comp. & Info Sciences

In this assignment, you need to write a Java program to use recursion and solve math puzzles like the following:

"Assume that variables T, H, E, B, S, P, and O denote distinct numerical values from 0 to 9 (inclusive). If THE specifies the decimal representation of a three-digit integer, BEST and SPOT specify decimal representations of two four digit integers, and SPOTS specifies the decimal representation of a five-digit integer, then, find the values of every variable T, H, E, B, S, P, and O so that the following addition operation becomes true: "

$$\begin{array}{cccccc} & & T & H & E \\ + & B & E & S & T \\ + & S & P & O & T \\ \hline S & P & O & T & S \end{array}$$

The main method of your program must first receive the puzzle from user through keyboard (say "THE+BEST+SPOT=SPOTS\n") and store the list of sum operands and the sum result in String objects (e.g. static ArrayList⟨String⟩ operands = ["THE", "BEST", "SPOT"] and static String result = "SPOTS"). Also, the list of all variables (letters occurring in operands or result) can be stored in another static variable (e.g. ArrayList⟨Character⟩ variables = ['T', 'H', 'E', 'B', 'S', 'P', 'O']). Finally, to solve the puzzle, the main method calls a static *recursive* method with the following signature that tries all possible combinations of values for the variables in this puzzle and find out which ones make the sum mathematically correct:

```
public static void solve(int[] digitValues)
```

The only input parameter of this method, int[] digitValues, is an array of integers initialized to new int[256] and is supposed to store the values of each variable (say if letter T represents digit 5, then digitValues['T'] = 5 where 'T' represents the ASCII code of letter T which is 84). For the first call to this method, digitValues has 256 cells[1], each are set to -1 (which represents an empty cell and means that no character is mapped to any digit yet). The base case condition of this recursive method can be set to when none of the puzzle variables are mapped to -1.

[1]Historically, ASCII table had 256 or 2^8 rows mapping 256 characters to integers 0, 1, ... 255.

How to Find the List of Variables

After the program receives a puzzle and stores it in a string (say String puzzle), we need to figure out the participating variables (letters) in the puzzle and store it in a list of characters. A suggestion is that you first define a static list of characters, like *static ArrayList⟨Character⟩*, and call the following method every time the user enters a puzzle:

```
private static void findAllVariables(String puzzle) {
    variables.clear();
    for(char c: puzzle.toCharArray())
        if(Character.isLetter(c) && !variables.contains(c))
            variables.add(c);
}
```

Figure 1: Given String puzzle, the method finds the list of unique characters and store it in the list of variables, which is stored statically.

How to Convert String to Number

To convert strings like "THE", "BEST", "SPOT", and "SPOTS" to their equivalent integer values, you may use the following method:

```
private static int numericalValue(String word, int[] digitValues) {
    int rv = 0;
    for(char c: word.toCharArray())
        if(digitValues[c] < 0)//if c is not assigned to any value
            return -1;//error
        else
            rv = 10 * rv + digitValues[c];
    return rv;
}
```

Figure 2: Java implementation of method "numericalValue"

As you see in this figure, the first parameter is a word representing a number, second parameter, digitValues, is an array of size 256, mapping each letter of the word to a digit from 0-9. If a cell of digitValues is equal to -1, it means that the mapping doesn't exist and method returns -1 as a sign of error. For example, if digitValues['T']= -1, it means that we haven't decided on the value of variable T yet. In this case, method returns -1.

Input/Output of the Program

The program receives the puzzles from keyboard: "THE+BEST+SPOT=SPOTS" followed by a new line. Please note that each line represents one puzzle. The output is the set of

ALL mappings of letters to digits that make the equation true. For example, the output for the given input is:

T: 9|H: 4|E: 3|B: 8|S: 1|P: 0|O: 2|

which makes the equation true: $943 + 8319 + 1029 = 10291$. Assuming that numbers can have leading zeros (i.e. T, B, and S can get value zero), then we have the following set of answers:

T: 3|H: 5|E: 4|B: 8|S: 0|P: 9|O: 7|
T: 4|H: 7|E: 2|B: 8|S: 0|P: 9|O: 6|
T: 6|H: 1|E: 8|B: 7|S: 0|P: 9|O: 3|
T: 6|H: 3|E: 8|B: 5|S: 0|P: 7|O: 1|
T: 6|H: 5|E: 8|B: 3|S: 0|P: 4|O: 9|
T: 8|H: 1|E: 4|B: 2|S: 0|P: 3|O: 5|
T: 8|H: 5|E: 4|B: 7|S: 0|P: 9|O: 1|
T: 8|H: 7|E: 4|B: 5|S: 0|P: 6|O: 9|
T: 9|H: 1|E: 2|B: 4|S: 0|P: 5|O: 6|
T: 9|H: 4|E: 3|B: 8|S: 1|P: 0|O: 2|

Grading Criteria and Submissions

If you print all mappings of letters to digits that make the equation true, you'll receive the full credit (100%). If you can exclude the mappings that map multiple letters to the same digit, you will receive 10% extra credit (110%). And finally, if you can exclude the mappings that assign zero to letters that are initial letters of the operands or the result, you will receive another 10% extra credit (120%).

You need to submit a *.zip* file compressing all .java files of the program. If you want to attempt the extra credit part, please submit a *readme.txt file* and **clearly state** in it that your code contains which of the extra credit part.