The CLP Program: Removing Human Error

Eddie Fernandez
Department of Computer Science, Furman University, Greenville, SC

Introduction

The CLP program has been a fundamental requirement for students seeking to obtain a degree from Furman University for decades, and the crediting system has undergone numerous evolutions. Today's system based on card swipes and paper slips can be deemed error free through the simple process of digitization.

Outline

By replacing the obsolete card readers currently used by the CLP program with ones that record date, time, AND the student ID value, a simple program can be written to parse the data, remove any values that don't rate a credit, and produce an easy to read file containing valid student information for crediting purposes.

Input

2/22/2016 17:25,99142521	2/22/2016 17:25,99142521
2/22/2016 17:24,83621924	2/22/2016 18:30,99142521
2/22/2016 17:27,97445821	2/22/2016 17:27,97445821
2/22/2016 17:28,100603723	2/22/2016 17:28,100603723
2/22/2016 18:33,100603723	2/22/2016 17:33,100603723
2/22/2016 17:23,97588221	2/22/2016 17:23,97588221
2/22/2016 18:30,97588221	2/22/2016 18:30,97588221
2/22/2016 17:27,97307521	2/22/2016 17:27,97307521
2/22/2016 18:37,97307521	2/22/2016 18:37,97307532
2/22/2016 17:28,97307521	2/22/2016 17:28,97307521
2/22/2016 18:33,97307521	2/22/2016 18:33,97307521
2/22/2016 17:30,94146721	2/22/2016 17:30,94146721
2/22/2016 18:35,94146721	2/22/2016 18:35,94146722

Here we have two comma delimited datasets used as input for the program. They are heavily modified versions of stored data from a laundry room door swipe on campus, made to simulate CLP card swipes. Column 2 is a partially modified version of Column 1 for error checking purposes. Seen here is a MM/DD/YYYY date format and a 24 hour time stamp, separated by a space, marking the date and time of the card swipe. After the comma is the numerical data of the student's ID, which is not identical to their student number.

Program

```
for (int i = 0; i < size - 1; i++) {
    if (idnumbers[i] == idnumbers[i + 1]) {
        if(isValid(times[i], times[i + 1]) == true) {
            validity[i] = true;
            validity[i+1] = true;
        } else {
            validity[i] = false;
            validity[i+1] = false;
        }
    } else if(idnumbers[i] != idnumbers[i +1]) {
        validity[i] = false;
    } else if(idnumbers[i+1] == idnumbers[i+2]) {
        if(isValid(times[i+1], times[i+2])) {
            validity[i+1] = true;
            validity[i+2] = true;
        } else{
            validity[i+1] = false;
            validity[i+2] = false;
        }
    }
}

//Triplecate checking
for (int i = 0; i < size - 2; i++) {
    for (int j = i + 1; j < size - 1; j++) {
        for (int k = j + 1; k < size; k++) {
            if (idnumbers[i] == idnumbers[j] && idnumbers[i] == idnumbers[k]) {
                validity[k] = false;
            }
        }
    }
}

if (idnumbers[size-1] == idnumbers[size - 2]) {
    if (isValid(times[size - 1], times[size - 2])){
        validity[size - 2] = true;
        validity[size - 1] = true;
    }
} else {
    validity[size - 2] = false;
    validity[size - 1] = false;
}
```

```
//Assigns the number to a slot in a spreadsheet, along with its date and time data
public void toSheet(int[] idnumbers, String[] times, boolean[] validity){
    //Determine method of acquire date and time data of each swipe from Registrar on Monday
    //If no date and time data is available, consider unique spreadsheets and numbering system...
    try {
        int size = idnumbers.length;
        //String output = "";
        @SuppressWarnings("resource")
        PrintStream out = new PrintStream("output.txt");


        for (int i = 0; i < size; i++) {

            if ( true) {
                out.printf("%d , %s, %s\n", idnumbers[i], times[i], validity[i] ?"true":"false");
                }
        }


        //generate whatever data you want
    } catch(IOException e) {
        e.printStackTrace();
    }
}

/*Determines the validity of card swipes by checking for matching pairs of numbers. If a pair

public boolean isValid(String timeOne, String timeTwo) {
    String[] units1 = timeOne.split(":");
    String[] units2 = timeTwo.split(":");
    int minutes1 = Integer.parseInt(units1[0]);
    int minutes2 = Integer.parseInt(units2[0]);
    int seconds1 = Integer.parseInt(units1[1]);
    int seconds2 = Integer.parseInt(units2[1]);

    int duration1 = 3600 * minutes1 + seconds1;
    int duration2 = 3600 * minutes2 + seconds2;

    if (duration2 - duration1 >= 2400){
        return true;
    }
    else{
        return false;
    }
}
```

Above are the three most vital pieces of the program: the validity check (left), and the output method (top right), and the time comparator (bottom right). Using these key methods, the program is able to compare matching numbers in the list, check the times of said numbers, and confirm whether they are valid or not. The output method then produces the image below.

Output

Output 1

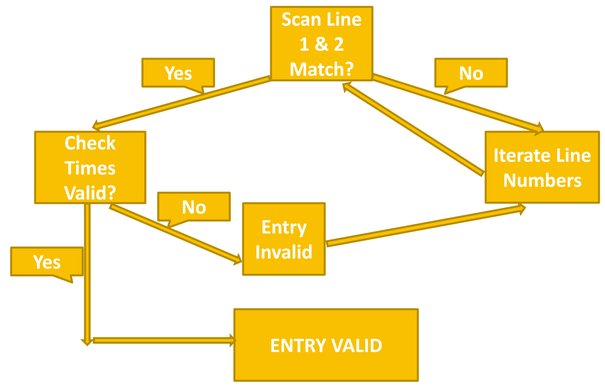
1	99142521 , 17:25, false
2	83621924 , 17:24, false
3	97445821 , 17:27, false
4	100603723 , 17:28, true
5	100603723 , 18:33, false
6	97588221 , 17:23, true
7	97588221 , 18:30, false
8	97307521 , 17:27, true
9	97307521 , 18:37, false
10	97307521 , 17:28, false
11	97307521 , 18:33, false
12	94146721 , 17:30, true
13	94146721 , 18:35, true

Output 2

1	99142521 , 17:25, true
2	99142521 , 18:30, false
3	97445821 , 17:27, false
4	100603723 , 17:28, false
5	100603723 , 17:33, false
6	97588221 , 17:23, true
7	97588221 , 18:30, false
8	97307521 , 17:27, false
9	97307532 , 18:37, false
10	97307521 , 17:28, true
11	97307521 , 18:33, false
12	94146721 , 17:30, false
13	94146722 , 18:35, false

What you see here are the output versions of the two input files to the left. Scanning line by line, the program looks for subsequent numbers that match, using a primitive algorithm. If the numbers match, their times are checked against each other, and if the difference is greater than a certain amount of time apart, the first entry is ruled true. The mate is only ruled false if there are student numbers following it. Otherwise, as seen in the first output, the validity ruling would match its mate. Using this data, the CLP creditor can easily enter the student numbers into the system for crediting purposes.

Design



The above algorithm is designed to handle door swipe styled data in sequential format. In the future, a simple modification to the algorithm involving a sort and search function could handle large, out of sequence data that would be produced by a two-swipe system.

That being said, the program, albeit a prototype, is designed to be modular in order to meet a varying need of automation. For instance, the program can have a web call method added to it that could interface with student records and ensure that the student is currently attending. The automation could encompass the whole system, where the CLP attendants plug the devices into a system and press GO.

Conclusion

While this program will likely not be adopted by the Registrar's office, it accomplishes it's mission by indicating need for a better system. The reliance on a physical medium, the legibility of the average student, and the current organizational methods leaves too much room for error in the system, and could result in vital CLP credits not being awarded to a present student. In the future, a better system will be introduced, but for now, the song remains the same.

Special Thanks

FURMAN COMPUTER SCIENCE

& Dr. Chris Healy