CSCI165 Computer Science II
Lab Assignment

Objectives:
- Read instructions carefully, ***all the way through*** (perhaps twice or thrice) ***before*** commencing with the clickity clack. For clarification, ask questions early.
- Review lecture and reading materials, including provided code examples
- Use data types correctly, applying conversions when necessary
- Apply math operations
- Process String data using API methods
- Implement a simple while loop and if statement
- Read/Write to external files

You have been provided with a text file containing the following data

- The name of various universities where an event will be held. The names will contain any number of spaces
- A ticket number for that event. Ticket numbers will vary in size
- The values are on separate lines but they are to be associated. Here is an example

  University of Minnesota – Morris (this is the venue)
  5167666 (this is a ticket number for that venue)

Problem Specification
Rabble Tickets Company sells tickets for educational talks, workshops, and other tech-related services. Because ticket agents frequently mistype long ticket numbers, Rabble Tickets has asked you to write an application that filters out invalid numbers.

Ticket numbers are designed so that if you drop the last digit of the number, then divide the number by 7, the remainder of the division will be identical to the dropped digit. This process is illustrated in the following example:

1. Examine a ticket number; for example, **123454**.
2. Remove the last digit, leaving **12345**. There are a couple of ways to approach this: math and String processing. Ultimately the value will need to be treated numerically due to the division that occurs.
3. Determine the remainder when the modified ticket number is divided by 7.
   1. In this case, **12345** divided by **7** leaves a remainder of **4** which does equal the dropped digit. **This ticket is valid and should be logged.**

Percentage: Your client also wants to keep track of the percentage of valid and invalid ticket numbers from any given data set. Calculate and display this value to the console (terminal) window formatted to two decimal places.

Program Flow:
1. Using a **command line argument**, pass in the name of the file to open and process. Open the file ***for input*** and open a new file ***for output.*** The output file should have the same name as the

input file with the token ***valid_*** as a prefix. For example: if the input file is called ***February_tickets.txt*** then the output file will be called ***valid_February_tickets.txt***

2. Using a while loop, iterate through the file, reading the venue names and ticket numbers
3. Perform the validation routine described above
4. Create a boolean variable to store the value of the ***comparison*** between the remainder and the digit dropped from the ticket number. Java uses if statements and = = for equality tests just like Python and C++. Call this variable **isValid**.

   boolean isValid = operand1 == operand2;

5. Use the variable described above (step 4) to determine if the ticket number should be logged. If it is valid, log the venue name and ticket number to the output file described above. Using the suggested naming convention above, this will make your if statement very readable:

   ```
   if (isValid){
     // write ticket and venue to a file
   }
   ```

6. The valid tickets numbers and associated venues should be written on the same line in the following format: (venue_name <colon> ticket number)

   venue_name: ticket_number

7. Get the next ticket number and repeat.

Given the provided text file, you should identify 101 valid tickets.

To reiterate the program flow:

- All code is written in **main**
- The program will ***read from one file*** and ***write to a different file***. This can all happen in the same while loop. You choose the methods of reading and writing. Multiple examples have been provided in the readings.
- You will need **try/catch** to handle the file opening. Do not forget to close both files when done.

The program deals with large amounts of potential ticket numbers at a time. You have no way of knowing how many ticket numbers are in a file, and the program needs to be able to adapt to a file ***of any size.*** The ticket number can be anywhere from 5 to 8 digits long and you have no way of knowing how long a given ticket number will be. The program need to be able to handle ticket numbers ***of any size.*** The goal of the problem is to filter out the valid ticket numbers from the invalid ticket numbers. The valid ticket numbers should be written to a file so that they can be distributed to customers.

I will grade your work with a different file of the same structure. The file will be of a different size with completely different venues and ticket numbers and will have a different name. You will not be given this file. There are no curve balls here.

Rubric

Requirement	**Points**
Included Detailed Comments including authorship, and descriptions of major logic sections	5
Code is correct and identifies the appropriate ticket numbers File names are correct Output format is per specification Correct percentage is calculated and displayed	20
Instructions were followed carefully	5