Course:	**INFO1214 Winter 2025 Project**
Professor:	**Bill Pulling**
Project:	***Lottery Prizes Analyzer Ver. 1.0***
Due Date:	**Code must be in the FOL drop box by Wednesday, April 9 by 2359 Eastern (11:59 PM)**
Service Pack:	**None yet. SP's are issued if any corrections need to be made.**

Background:

The Cash Prizes Abound (CPA) Lottery Corporation sells tickets for a variety of lotteries. Each lottery ticket has a series of numbers (non-negative integers). How many numbers appear in this series and their range of values depends on which lottery the ticket is for. For example, a ticket for the *LottaCash* lottery has a series of six numbers ranging in value from 1 to 19 whereas a *FabFive* lottery ticket has only five numbers between 5 and 55. When a customer buys a ticket, regardless of which lottery it's for, they can either choose their numbers themselves or they can let the CPA Lottery ticket system choose numbers randomly for them(called a 'quick pick') at the point of sale. On the day of the big draw, a series of completely random numbers are selected by CPA as the winning numbers and cash prizes are awarded to ticket holders as follows:

Prize	# of Matching Numbers	% of Prize Pool*
Grand Prize	All	85
Second Prize	All but one	7
Third Prize	All but two	8

* NOTE: there can be more than one winning ticket for any prize. In this case the total money awarded will be divided equal between the holders of the winning tickets. *(Note: Third prize pool is a greater percent of the prize money pool than the second prize pool because it is statistically more probable that a much larger number of ticket holders will have to split that prize.)*

Your job is to analyze the ticket numbers data contained in the provided data file to determine which ticket holders have won a prize, how many ticket holders will have to share a prize and how much money they'll each receive.

Input File Format

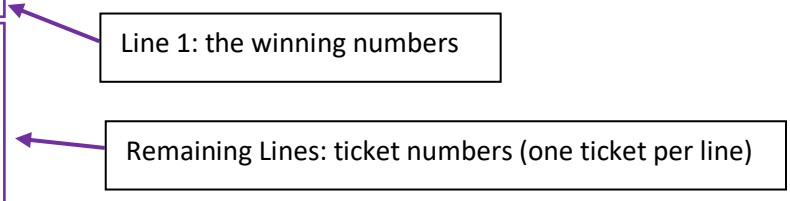
Any suitable input file for your program will be formatted as a comma separated Excel file (CSV). This means that all fields contained on a line of data are separated by commas and each line will end with an end-of-line character.

The first line will contain the winning numbers consisting of a series of positive integers, comma-separated and in ascending order. How many numbers there are in the series and their exact value range depends on which lottery the data is for (LottaCash uses 6 numbers, but the FabFive lottery uses only 5). However, the numbers will always be within a maximum range of 1 to 99 inclusive. Furthermore, there will only be between 4 and 10 numbers in a series and all series (lines) in a data file will have the same number of values.

Each line that follows the first line represents a ticket from a purchased ticket. There can be any number of tickets sold (even zero!) so there could be one or more lines in a data file.

As an example, here are the first few rows of a sample data file for the LottaCash lottery:

```
3,4,5,6,14,16
5,8,9,11,16,17
1,4,10,16,17,18
1,2,6,10,11,13
2,6,12,14,16,18
3,4,5,10,13,18
1,4,6,10,16,17
```



<u>Important</u>: Your program should be able to analyze any data file that matches this format(winning numbers on first line, ticket numbers on the remaining lines) regardless of how many lines it contains or how many numbers appear on a line (up to 10). So, if I test your code with a file containing LottoMax tickets(7 numbers from 1 to 49) then your code should be able to analyze it and produce the correct output. When you are testing your code you will need to make up some short comma separated test files with different amounts of numbers to test that your code can do this.

General Structure of Your Program

Your class containing the main method should be called ***Your_Initials*_LotteryPrizes.java**. This will contain the high level logic for your program (generating output, obtaining user inputs, high level processing of the ticket number, etc.).

However, we want you to reduce the amount of code that appears in your main method of your source code . For this reason, some lower-level operational code will be done inside methods that will be stored in a helper class called ***Your_Initials*_ProjectMethods.java**. These methods will be called from the main method and are outlined in the steps below.

Step 1: Your program needs to ask the user to input the following three pieces of information:

- A description or name for the lottery (to be included in the program's output or report)
- The total amount of money in the prize pool. No decimal places are required, we want dollars only and no cents. It should be validated to ensure that the prize pool is at least $1,000.
- The path and name for the input file (e.g. "lottery.csv")

Step 2: Your program needs to set-up a Scanner object for reading data from the file. This should include suitable exception handling code so that no runtime error occurs if the file is not found. It's acceptable to have the program display an informative error message such as "Could not find the file, please check path name and try again." and then exit gracefully if this is the case.

A NOTE ABOUT READING FROM THE FILE: When working with a large amount of data, some programmers make the mistake of assuming their program needs to load the entire data set into a data structure. This is often not necessary and consumes large amounts of RAM which can make the program run slower. For this project your program needs to keep the winning numbers in a data structure for comparison (the first line of the file) and then hold only one set of ticket numbers at a time (another line) for processing. The only other exception to this is that when any winning ticket is identified, it can be stored in another data structure so that it can be displayed in a final report.

Step 3: You must write a class method called ***getNextSeries()***. This method will accept the initialized Scanner object that's associated with the file and return an array of integers which is the series of numbers contained in the next line of the file. This means that it needs to split-up the tokens contained in the line based on where the comma delimiters are. It also needs to create a suitably-sized array of integers, convert the tokens into integers and assign each integer to an element of the array before returning the array. Note that the very first time this method is called it will return the winning numbers. Each time it's called after that it will return the numbers from another customer's ticket. The method may assume that there is another line of data to be read from the file. It should only be called by your main method if this is the case.

Step 4: Write another class method called ***countMatchingNumbers()*** which accepts two integer arrays as arguments. The first will contain the numbers for one ticket and the second will contain the winning numbers. You'll need to come up with an algorithm to compare the numbers in each array to count the number of matches and return this count (an integer) from the method. The two arrays will contain the same number of elements, but the actual number of elements depends on which data file is currently being processed. Keep the following two things in mind when designing your algorithm:

- The integers in each array will already be sorted from lowest to highest value.
- NOTE: You can't simply compare elements that are at the same index in each array. To understand why, look at the following example which matches all but two of the winning numbers making it a third prize winner.

Array of ticket numbers	
Index	Number
0	2
1	7
2	18
3	29
4	44
5	47

Array of winning numbers	
Index	Number
0	7
1	16
2	18
3	30
4	29
5	44

Step 5: Create a third called method, called ***formatTicketNumbers()***, which will accept an integer array containing one set of ticket numbers and return a formatted string containing the numbers. The format should incorporate the numbers in order on the same line, delimited somehow. Here's one example that uses a comma and a space as the delimiters: "2, 7, 18, 29, 44, 47".

Step 6: Now in your main() method, set-up two data structures to store the numbers for tickets that have won a prize. You'll need two separate data structures: one for second prize winners and one for third prize winners. *(You won't need one for grand prize winners because these tickets will exactly match the winning numbers. However, you should set-up a counter variable to keep track of the number of tickets that are grand prize winners.)* Keep in mind that there could be any number of second prize or third prize winning tickets. This means these data structures should be expandable so that an element can be added each time another winning ticket is found. Each element of these data structures will represent an array of integers. Here are two options for how you can do this:

- Create data structures where each element is an integer array, or…
- Create data structures where each element is a string and then convert an array of ticket numbers to a formatted string by calling the *formatTicketNumbers()* method before storing it in the data structure

Step 7: Continuing with the main() method, call the *getNextSeries()* method once to obtain and store the winning numbers. Then, set-up a loop that will call the method again repeatedly until there are no more lines to be read in the file. Each iteration of this loop should test the current set of ticket numbers against the winning numbers using the *countMatchingNumbers()* method to determine of it's a winner. Remember that a ticket can be a winning ticket in three ways:

- Grand prize winner if all numbers match the winning numbers
- Second prize winner if all but one of the numbers match the winning numbers
- Third prize winner if all but two of the numbers match the winning numbers

If a ticket is a grand prize winner, increment the "grand prize winner" counter to keep track of how many winners will be splitting the grand prize. If a ticket is a second prize or third prize winner, save the ticket numbers to the corresponding data structure you created in step 6 for this purpose.

Step 8: After analyzing all the ticket numbers in the file, your main() method should generate a report showing the results of this analysis, including:

- The name or description of the lottery as entered by the user
- The total prize pool ($)
- The number of tickets purchased and analyzed
- The winning numbers (from the first line of the file)
- The number of tickets that won the grand prize
- The percentage of the prize pool allocated for the grand prize
- The total value of the grand prize ($)
- The value of the grand prize per winning ticket ($)

- The number of tickets that won the second prize
- The number of matching numbers required to win second prize
- The percentage of the prize pool allocated for a second prize
- The total value of the second prize ($)
- The value of the second prize per winning ticket ($)
- A list of ticket numbers that won the second prize
- The number of tickets that won the third prize
- The number of matching numbers required to win third prize
- The percentage of the prize pool allocated for the third prize
- The total value of the third prize ($)
- The value of the third prize per winning ticket ($)
- A list of ticket numbers that won the third prize

All output should be formatted approximately as shown in the following sample output. If you use a different format it must be well organized, easy to read and all dollar figures must include the '$' symbol, the ',' separator and show either 0 or 2 decimal places depending on which is appropriate (as demonstrated below). Note that the heading for the second prize says "Second prize winners (5 numbers match)" should be automatically adjusted so that it is accurate for the data file used. For example, if the data includes 7 numbers on a ticket, the heading should state "Second prize winners (6 numbers match)". This is also true for the third prize heading which would state "Third prize winners (5 numbers match)".

Sample Output: *(based on the file 'lottacash.csv' which is posted for you)*

```
Lottery Prizes Analyzer

Enter the name of the lottery: LottaCash
Enter the amount of money in the prize pool: $10000
Enter the path for the data file: lottacash.csv

Lottery Prizes Report
---------------------

Lottery Name:        LottaCash
Total prize pool:    $10,000
Number of tickets:   1,000
Winning numbers:     2, 9, 12, 14, 15, 18

Grand prize winners (all numbers match)...
  Number of winners: 1
  % of prize pool:   85.0
  Total prize value: $8,500.00
  Prize per ticket:  $8,500.00

Second prize winners (5 numbers match)...
  Number of winners: 2
  % of prize pool:   7.0
```

```
  Total prize value: $700.00
  Prize per ticket:  $350.00
  Ticket Numbers:    1, 2, 9, 12, 14, 18     2, 9, 12, 14, 15, 17

Third prize winners (4 numbers match)...
  Number of winners: 41
  % of prize pool:   8.0
  Total prize value: $800.00
  Prize per ticket:  $19.51
  Ticket Numbers:    2, 5, 9, 11, 12, 18     2, 4, 12, 15, 16, 18
                     9, 10, 14, 15, 17, 18   2, 6, 11, 12, 15, 18
                     2, 11, 12, 14, 16, 18   5, 6, 9, 12, 14, 18
                     6, 11, 12, 14, 15, 18   1, 2, 9, 12, 14, 17
                     2, 6, 9, 13, 14, 15     2, 3, 5, 14, 15, 18
                     2, 5, 6, 12, 14, 15     2, 4, 9, 12, 17, 18
                     3, 9, 10, 12, 14, 18    2, 4, 6, 12, 14, 15
                     2, 6, 12, 14, 18, 19    2, 3, 5, 9, 15, 18
                     1, 9, 10, 12, 15, 18    5, 8, 9, 14, 15, 18
                     1, 2, 6, 12, 14, 18     2, 4, 5, 9, 15, 18
                     9, 12, 13, 14, 18, 19   6, 9, 14, 15, 17, 18
                     1, 7, 9, 12, 14, 15     2, 11, 12, 13, 14, 18
                     6, 7, 12, 14, 15, 18    7, 12, 14, 15, 18, 19
                     2, 3, 9, 15, 18, 19     2, 4, 9, 15, 16, 18
                     1, 2, 12, 14, 17, 18    2, 3, 7, 14, 15, 18
                     3, 7, 9, 14, 15, 18     2, 4, 9, 12, 14, 16
                     2, 10, 12, 13, 14, 15   6, 7, 12, 14, 15, 18
                     1, 2, 12, 13, 14, 18    9, 14, 15, 17, 18, 19
                     2, 8, 9, 14, 15, 19     4, 9, 14, 15, 16, 18
                     2, 4, 5, 12, 14, 18     2, 4, 10, 12, 14, 18
                     3, 12, 14, 15, 18, 19
```

***** BONUS OPTION *****

NOTE: YOU CAN SCORE THE BONUS MARKS ***ONLY IF ALL OF THE BASIC REQUIREMENTS OF THE PROJECT ARE MET AND ARE COMPLETELY FUNCTIONAL***. YOU CANNOT USE THE BONUS REQUIREMENT TO MAKE UP FOR MARKS LOST FOR MISSING THE BASIC FUNCTIONALITY.

FULFILL THE BASIC REQUIREMENTS COMPLETELY BEFORE EVEN THINKING ABOUT DOING THE BONUS.

If you are up to the challenge, there are an additional three bonus marks available.

You must arrange the lists of second and third prize-winning tickets so that any duplicate tickets are identified (i.e. identical lines in the input file that are winners) and are only listed once, followed by a set of parentheses holding the number of tickets that had the same set of numbers.

For example:

1, 20, 26, 28, 31, 33 (3)

states that there were three winning tickets with the exact numbers 1, 20, 26, 28, 31, 33.

Make sure the ticket count for each prize level is still correct after adjusting your logic to accommodate this change.

Submitting your work:

- Zip up ALL OF YOUR .java SOURCE FILES (main class and the helper class) and submit them to the Project submission folder in FOL by the deadline indicated on page 1 of this document. NOTE THAT YOU ONLY GET ONE SUBMISSION ATTEMPT, SO TEST YOUR CODE THOROUGHLY.
- You do not need to submit the data file, as I have a copy of it to use.

Submit your project on time!

- Code submissions to the FOL drop box must be made on time! Late projects will be subject to late penalties of 10% per day up to a maximum of 5 days. After 5 days a mark of zero will be given.

Submit your own work!

- It's considered cheating to submit work done by another student or from another source such as ChatGPT as your own work. This is called "plagiarism". Helping another student cheat by letting them copy your source code files is called "abetting plagiarism". Both are academic offences.
- YOU MUST WRITE YOUR OWN CODE!
  Students are encouraged to share ideas and to work together on practice exercises, and you can help someone else to fix a bug in their code, but any code or documentation prepared for your project must be done by you. Penalties for committing plagiarism, or helping another student plagiarize by sharing source code with them, include a zero grade and an academic offence being filed against the student or students involved. All submissions will be analyzed using plagiarism and AI detection software especially designed to analyze Java code.

Marking Key:

	Marks Available	Marks Assigned
STYLE: Appropriate pseudo-code statements are done and program is well documented with comments, including a complete documentation header. Program compiles without errors and runs without crashing. Naming conventions followed, variable, constant, and method names are all representative of what each does. No compiler warnings for unused variables or unclosed objects.	2	
Basic Requirements		
Use inputs are obtained with user prompts and assigned to appropriate variables. Data validation is done on the prize pool.	1	
FILE HANDLING: File is successfully opened where possible and exception handling is used to prevent a runtime error when the file can't be found.	2	
METHOD *getNextSeries()*: • class method defined in the helper class • fully documented • accepts a Scanner object • returns an array populated with all integers from the next line of the file	3	
METHOD *countMatchingNumbers()* : • class method defined in the helper class • fully documented • accepts two int arrays • returns the number of matching numbers that exist in both arrays	3	
METHOD *formatTicketNumbers()* : • class method defined in the helper class • fully documented • accepts an int array • returns a formatted String containing the numbers in the array	3	
Main program processes all ticket numbers in the data file to generate: • a count of the grand prize winners • a data structure of all the ticket numbers of second prize winners • a data structure of all the ticket numbers of third prize winners NOTE: Full marks will be awarded for this project only if required methods are used to process the lottery data.	3	
A REPORT is generated with all required information including correctly calculated prize amounts and is 100% correct even when using different data files with different numbers of lines and a different number of integers per ticket. Required format is used for dollar amounts. The report is well organized and easy to read.	3	
TOTAL	20	
Bonus Requirements		
BONUS: List of winning tickets (for 2nd & 3rd prizes) only displays duplicate ticket numbers once and reports the number of tickets having the same set of numbers. Winning tickets count shown is still correct! **NOTE: all items above must be done and completely functional before marks will be awarded for this bonus option**.	3	