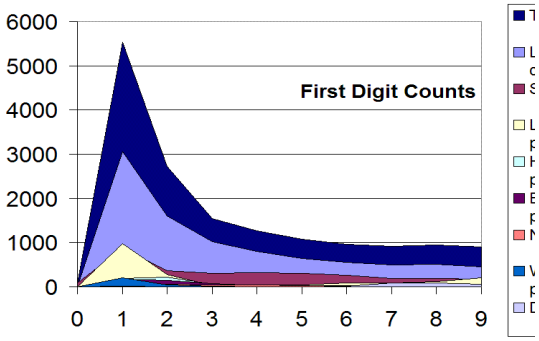
CS 202 – Assignment #1
Spring 2022

Purpose: Review basic C++ language constructs and file I/O.
Points: 100

Video Link for Assignment1:
https://unlv.webex.com/unlv/ldr.php?RCID=f8b26197319626b2d3bed321fc80d44b

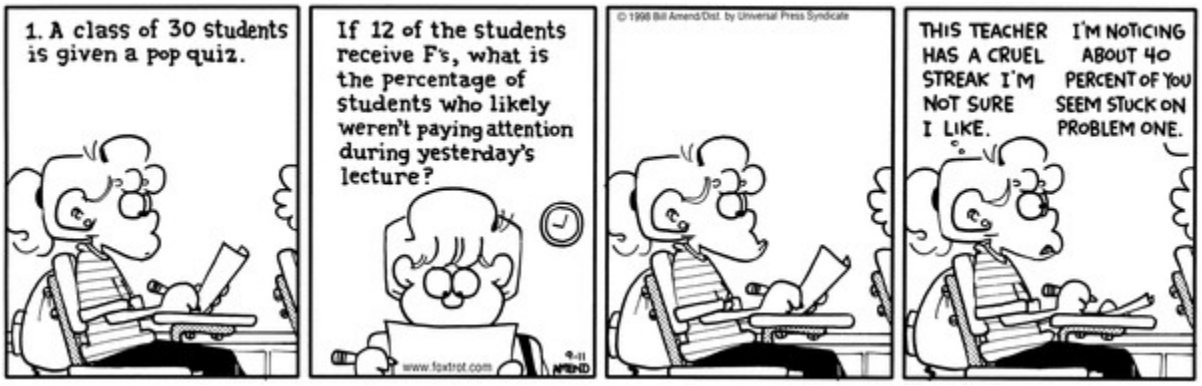
Assignment:
In this assignment, you will write a program that determines the distribution of initial digits in a set of data. Benford's Law[1] that describes a surprising pattern in the frequency of occurrence of the digits 1–9 as the first digits of natural data. You might expect each digit to occur with equal frequency in arbitrary data. Indeed, in truly random data (over appropriate ranges) each digit does appear with equal frequency. However, a substantial amount of natural data from sweepingly diverse sources does *not* exhibit a uniform distribution. Instead, 1 is more common than 2, 2 more common than 3, and so forth. For example:

We will test this law using a program to read data sources and count the leading digits. Write a C++ program to read a data file containing a series of numbers. The data file must contain "real world" or "natural" data. Several examples will be supplied, but you may obtain your own data sets. The first five (5) lines of the file will contain header and data source information and should always be skipped by the program.



The program should read the file name from the command line, read the name, open the file, and read the numbers. The program should determine the 1st digit of the number and count the occurrences. You can obtain the 1st digit by repeated division by 10. Or (for math majors), there is a tricky solution using logarithms that avoids the repeated division. The program should display the results, including a small graph of the results (see example). The results should be written to a file (in addition to the console).

If the file can not be opened, the program should display an appropriate error message and terminate.



1 For more information, refer to: http://en.wikipedia.org/wiki/Benford's_law

Function Descriptions:
The following are more detailed descriptions of the required functions.

- `int findFirstDigit(int number)`- This function has one parameter. The function is to find the FirstDigit of the number and return that first digit.
- `int findHundreds(ofstream &resultFile, double value)`- This function has two parameters. This function is to determine the number of hundreds from the given value(2nd parameter) and it have to print the asterisks(stars) accordingly in the resultsFile and also return that number(hundreds).
  For example, if the value passed to this function is 1606, then 1606/100=16.06 and so the function should print 16 asterisks and return the value 16. For example, if the value is 3056, then 3056/100=30.5, and so the function should print 31 asterisks and return the value 31.(If the first digit after the decimal point is greater than 4, then it should round the value to the next integer) (Hint: Use round(), setfill('*'), setw(), etc.)

Linking Instructions:
A **make** file is provided. As projects get larger, compiling a series of source files by hand becomes tedious. Make is a utility that will read a make file (named **makefile** by default), compile applicable source(s), and build the executable file. Assuming the main file is named **`benfordsLaw.cpp`**, the following are the instructions to build the executable.

```
kishore-vm% make
```

Which will create an executable named **benfordsLaw**. The provided make file assumes these file names. As such, you should not change the file names for this assignment.

Program Header Block
All program and header files must include your name, section number, assignment, NSHE number, input and output summary. The required format is as follows:

```
/*
Name: MY_NAME, NSHE, CLASS-SECTION, ASSIGNMENT
Description: <per assignment>
Input: <per assignment>
Output: <per assignment>
*/
```

Failure to include your name in this format will result in a loss of up to 5%.

Code Quality Checks
A C++ linter[2] is used to perform some basic checks on code quality. These checks include, but are not limited to, the following:

- Unnecessary 'else' statements should not be used.
- Identifier naming style should be either camelCase or snake_case (consistently chosen).
- Named constants should be used in place of literals.
- Correct indentation should be used.
- Redundant return/continue statements should not be used.
- Selection conditions should be in the simplest form possible.
- Function prototypes should be free of top level *const*.

Not all of these items will apply to every program. Failure to to address these guidelines will result in a loss of up to 5%.

2 For more information, refer to: https://en.wikipedia.org/wiki/Lint_(software)

Scoring Rubric
Scoring will include functionality, code quality, and documentation. Below is a summary of the scoring rubric for this assignment.

Criteria	Weight	Summary
Compilation	-	Failure to compile will result in a score of 0.
Program Header	5%	Must include header block in the required format (see above).
General Comments	10%	Must include an appropriate level of program documentation.
Line Length	5%	No lines should exceed more than eighty (80) characters.
Code Quality	5%	Must meet some basic code quality checks (see above)
Program Functionality (and on-time)	75%	Program must meet the functional requirements as outlined in the assignment. Must be submitted on time for full score.

Steps you can follow:
These steps will help/guide you to complete this assignment, but are not limited to, the following:

- This assignment use command line arguments, file streams, functions, etc.
- Skeleton code (benfordsLaw.cpp) and makefile is provided along with 4 text files.
- Declare the variables and initialize as needed for the program.
- Once you compile using make, to run the code you have to pass the filename as the second argument.
  `./benfordsLaw librarybook.txt`
  If the second argument or filename is invalid, then your program should re-prompt to enter the filename until it is successful. Also, it should print an error message-
  `"Error: You entered " << filename << " \nplease re-enter valid filename:"`
  Make sure that any filename other than given librarybooks.txt, livejournal.txt, random.txt, sunspots.txt, should be considered as invalid filename.
- If the number of arguments are less than 2 or greater than 2, then it should print an error message `- Usage: ./benFordsLaw <base filename>.txt`
- You will write the code in the given benfordsLaw.cpp and you will submit only this file in codegrade. (Do not submit other files)
- The main agenda of this assignment is to read the numbers(int) from each file and determine the first digits of each number and keep a count of each digit. For example,
  First Digit:

  ```
  0       0|-0
  1    3056|******************************-31
  2    1606|****************-16
  ...
  ```
- Each input file has 5 lines of header comments, you should write a logic to skip those lines.

- From 6th line to the end of file,
  data - 12201
  data - 600778
  …
  You can use your own logic to read the numbers(int) from the file and update the firstDigit count. (To keep a count of each digit you can use if/else-if ladder or switch case)
- In this assignment, you have to save the result to an output file only. (Do not print anything on console window)
- You have to save the results to a different file and your output filename should follow this- For example, if the input file name is librarybooks.txt, then the output filename should be librarybooks_results.txt. Similarly, you will get 4 output files in total.
- Please check the Example execution for more details on formatting your code. Total data points in example execution is the total count of all first digits in the given file.
- Do not add additional functions apart from the given two functions.

Submission:

- Just submit your benfordsLaw.cpp only.

- **Things to Remember**
- All files must compile and execute on Ubuntu and compile with C++11.
- Submit source files
  - Submit a copy of the source file via the on-line submission
- Once you submit, the system will score the project and provide feedback.
  - If you do not get full score, you can (and should) correct and resubmit.
  - You can re-submit an unlimited number of times before the due date/time.
- Late submissions will be accepted for a period of 24 hours after the due date/time for any given lab. Late submissions will be subject to a ~2% reduction in points per an hour late. If you submit 1 minute - 1 hour late -2%, 1-2 hours late -4%, … , 23-24 hours late -50%. This means after 24 hours late submissions will receive an automatic 0.
- **Check the Example Execution on next page**

Example Execution:
Below is an example program execution.

```
Kishore-vm:% make
kishore-vm:% ./benfordsLaw
Usage: ./benFordsLaw <base filename>.txt
Kishore-vm:%./benFordsLaw check files
Usage: ./benFordsLaw <base filename>.txt
kishore-vm:% ./benfordsLaw hat.txt
Error: You entered hat.txt
please re-enter valid filename:
librarybooks.txt
Kishore-vm:% cat librarybooks_results.txt

CS 202 - Assignment #1
Benford's Law Program
File Name: librarybooks.txt

---------------------------------
Benford's Law - Test Results:
Total Data Points: 9138
First Digit:

0       0|-0
1    3056|******************************-31
2    1606|****************-16
3    1018|**********-10
4     801|********-8
5     640|******-6
6     560|******-6
7     502|*****-5
8     503|*****-5
9     452|*****-5
---------------------------------

Note: For more detailed output, check "CS202_Assignment1_Example-
Execution.rtf" or "CS202_Assignment1_Sample-Execution.txt"
```