YORK UNIVERSITY
LASSONDE SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

EECS 2032E - Introduction to Embedded Systems

Summer 2025

Lab 3

Due Date: May 31, 2025

Lab Objectives

- To write simple bash scripts using conditional statements, loops, and functions.

Note:

- The red text is what you type and the blue text is the computer response.
- Save the `.sh` files in a folder named Lab3. Compress the folder into a ZIP file and submit it on eClass.

Pre-Lab

Review the course slides of Week 3. You may need to refer the course slides of Week 1 and 2, the probability is small though.

Problem 1

Write a bash script, named `lab3_1.sh`, that reads two numbers on the same line (no prompt). If you read one or zero numbers, it displays: `You should enter two numbers!`. If the numbers are equal it displays: `These two numbers are equal!`. If the two numbers are not equal, then the script checks if the larger number can be perfectly divided by the smaller one. If it can, it displays: `big_num is n times the small_num!`, where `big_num` is the larger number, `small_num` number is the smaller one, and n is the ratio (factor) by which the smaller number fits into the larger one. If it cannot, it simply displays: `No relation!`.

Problem 2

Write a bash script, named `lab3_2.sh`, that implements a number guessing game using two functions:

- `generate_random_number()`: To generate a random number between 1 and 100
- `play_game()`: To handle the main logic for a number guessing game

Upon calling `play_game()`, the game gets started. A number is first selected randomly via `generate_random_number()`. The user is then prompted to enter a positive integer. The script should first validate the input to ensure it matches the criteria of being a positive integer (i.e., it should only consist of digits). If the input is invalid, the script should prompt the user to enter a valid positive integer again. If the input is valid, it provides the feedback on whether the guess is too high, too low, or correct. The script also keeps track of the number of attempts made by the user.

Note: The script should allow the user to input their guesses until they either guess the correct number or decide to exit the game by typing `exit`.

Hints

- You may use the regular expression (regex) `^[0-9]+$` to validate the input (Here, `^` asserts the start of a line or string, `[0-9]` mathces any single digit between 0 and 9, `+` means one or more of preceding elements, and `$` asserts the end of a line or string)
- You may make use of the variable `$RANDOM`, which is a built-in variable to generate a random integer each time it is referenced

```
$ bash lab3_2.sh
Welcome to the Number Guessing Game!  I have selected a number between 1
and 100.  Can you guess it? Enter your guess (or type 'exit' to quit):  4.5
Please enter a valid number.
Enter your guess (or type 'exit' to quit):  8
Too low!  Try again.
Enter your guess (or type 'exit' to quit):  15
Too high!  Try again.
Enter your guess (or type 'exit' to quit):  10
Congratulations!  You've guessed the number 10 in 3 attempts.
```

```
$ bash lab3_2.sh
Welcome to the Number Guessing Game!  I have selected a number between 1
and 100.  Can you guess it? Enter your guess (or type 'exit' to quit):  4.5
Please enter a valid number.
Enter your guess (or type 'exit' to quit):  8
Too low!  Try again.
Enter your guess (or type 'exit' to quit):  15
Too high!  Try again.
Enter your guess (or type 'exit' to quit):  exit
Thank you for playing!  The number was:  10
```

Problem 3

A small library system consists of two files:

- `books.txt`: Contains record with the fields, Book ID, Book title, Author, Number of copies available; the fields are separated by `:`. Following is an example:

  101:The Great Gatsby:F. Scott Fitzgerals:3

- `borrowed.txt`: Contains records of borrowed books with two fields, Book ID and the Number of copies borrowed; the fields are separated by `:`. Following is an example:

  101:2

Write a bash script that checks these two files and outputs the book title of any book where the number of copies available is less than the number of copies borrowed.
Save the file as `lab3_3.sh`

Hints

- You may find the use of *associative* arrays (declared as `declare -A`), which allows to use strings as keys (unique identifiers that are used to access the corresponding values stored in the array), which is very useful for mapping book IDs to borrowed counts

Optional

- Consider sorting the output by book title or another criterion for better readability
- Break down the script into functions for better organization and reusability

Problem 4

Write a bash script that reads a list of words from the user (separated by spaces) and displays:

1. The word that appears most frequently
2. The number of times that word appears

Hint: You may consider using an *associative* array (declared as `declare -A`).

Example

```
Enter a list of words separated by spaces: apple banana apple orange banana apple
Most frequent word:  apple
Occurrences:  3
```

Save the file as `lab3_4.sh`