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

EECS 2032E - Introduction to Embedded Systems

Summer 2025

Lab 4

Due Date: June 7, 2025

Lab Objectives

- To write `sed` commands for text processing in `bash`.
- To get familiar with writing and compiling `C` programs.

Note:

- This lab should be easy and takes little time to complete.
- The red text is what you type and the blue text is the computer response.
- Save the `.sed` files and `.c` files in a folder named Lab4. Compress the folder into a ZIP file and submit it on eClass.

Pre-Lab

Review the course slides of Week 4.

Problem 1

This problem has two parts:

a) Write a `sed` command that changes the phone number format in the file, named `phonesA.txt`, as follows:
(area_code)prefix-number → area_code-prefix-number
For example, (123)456-7890 → 123-456-7890

Save the file as `lab4_1.sed`

b) Write a `sed` command to do the opposite, that is, changes the phone number format in the file, named `phonesB.txt` as follows:
area_code-prefix-number → (area_code)prefix-number
For example, 123-456-7890 → (123)456-7890

Save the file as `lab4_2.sed`

Problem 2

Write a `C` code to read one integer (M) followed by another two integers (i, j). Display the ith through the jth digits of M.

For example if the input is:

1298567 1 4

It should display digits 1 through 4,i.e., display 9856

Integer	1	2	9	8	5	6	7
Digit Number	6	5	4	3	2	1	0

Hints:

- Read as integer, not string
- Use a combination of integer division and modulus operators to get the result
- For example, in the above case, first you divide the number by 10^x, in that case $x = 1$ to get 129856
- Then use modulus operator with 10^y ($y = 4$) as a second operand to get 9856
- Your job is to calculate the value of x and y and their relation to i and j; you may use a loop that multiply by 10 to get the power of 10 you need

Save the file as `lab4_3.c`

Problem 3

In embedded systems, we often read noisy sensor data and need to process it in real time. One simple and powerful technique is a *moving average filter*, which smooths out the data. Write a `C` program that simulates 10 consecutive readings from a sensor (you can use user input). The program should compute a moving average over the last `N` readings (say, `N = 4`) and print it every time a new value is added.

Expected Behavior:

```
Enter reading 1:10
Average:  10.00

Enter reading 2:14
Average:  12.00

Enter reading 3:12
Average:  12.00

Enter reading 4:8
Average:  11.00

Enter reading 5:10
Average:  11.00

Enter reading 5:16
Average:  11.50
...
```

Save the file as `lab4_4.c`