Nicholas LaJoie, ECE 331, HW 3

```
// Author: Nicholas LaJoie
// ECE 331 - Homework 3
// Date: 2/9/17

1. Linux Kernel preparation commands:
    a. sudo apt-get update
    b. cd /usr/src/
       sudo git clone https://github.com/raspberrypi/linux.git
       sudo chown pi linux/
    c. uname -r
       git log
       (search) /4\.4\.35 (we need the commit just before this one)
       git checkout 1ba7fafae3c2c1bcafa838a36db0fd358edb18af
    d. sudo mv /usr/src/linux/ /usr/src/linux-4.4.34
    e. make mrproper
       sudo modprobe configs
       gunzip -c /proc/config.gz > .config
       ln -s /boot/Module7.symvers Module.symvers
       make modules_prepare
    f. cd /lib/modules/4.4.34-v7+
       sudo ln -s /usr/src/linux-4.4.34/ build
    g. sudo ln -s build source

2. Perl Style Reg-Expression:
    a. Minimum Match: .0
    b. [-+]?[0-9]*\.?[0-9]+([eE][-+]?[0-9]+)?


3. Reg-Expression Program Source Code:

// Author: Nicholas LaJoie
// ECE 331 - Homework 3
// Date: 2/9/17
// Description: Accepts a regular expression via stdin and prints entire lines that contain
 any matches
// to stdout. Uses regcomp() and regexec()
// Sources: Valuable information obtained from linux.die.net/man/3/regcomp
//          Use of getline: c-for-dummies.com/blog/?p=1112
//          Removal of newline char: stackoverflow.com/questions/2693776/removing-trailing-
newline-character
//          -from-fgets-input
//          Use of regex: stackoverflow.com/questions/1085083/regular-expressions-in-c-exam
ples
//          Reading from stdin: stackoverflow.com/questions/22340845/c-piping-file-from-com
mand-line-to-c-program-with-the-use-of-strtok

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <regex.h>

void rm_newline(char* str);

int main(int argc, char * argv[])
{
    char *line;
    size_t len = 0;

    regex_t reg;
    int regflag;
    char errbuf[100];

    // Validate proper command line argument
    if (argc != 2) {
        perror("Invalid arguments.\n");
            return 1;
    }
```

```
    // Load Regular Expression
    regflag = regcomp(&reg, argv[1], REG_EXTENDED);
    if (regflag) {
        perror("Error compiling regex\n");
        return 2;
    }
    // Allocate input line memory
    line = (char *) malloc(len * sizeof(char));
    if (line == NULL) {
        perror("Error allocating line memory.\n");
        return 3;
    }
    // Read user input using stdin
    while (getline(&line, &len, stdin) != -1) {
        // Remove newline character at the end of the line
        rm_newline(line);
        // Process line using regex
        regflag = regexec(&reg, line, 0, NULL, 0);
        if (!regflag) {
            printf("%s\n", line);
        } else if (regflag == REG_NOMATCH) {
            continue;
        } else {
            regerror(regflag, &reg, errbuf, sizeof(errbuf));
            printf("Regex error: %s\n", errbuf);
            return 4;
        }
    }
    free(line);

    return 0;
}

void rm_newline(char* str)
{
    int len = strlen(str);

    if (len > 0 && str[len - 1] == '\n') {
        str[--len] = '\0';
    }
}
```

4. Automation testing for Prob. 3 Source Code:

To test the following source code, run
`>> cat test_prob3 | ./prob3 '[-+]?[0-9]*\.?[0-9]+([eE][-+]?[0-9]+)?'`

```
// Author: Nicholas LaJoie
// ECE 331 - HW 3, Problem 4
// Description: File provides various input to test problem 3 source code
//              Code should print lines that contain a C-style floating point number

A bunch of floating point style numbers that should match:
123.123
1e0
1e1
-1e1
-1.e0
-2.3e7
-3.2e-9
-1.
-.1
+1.
+.1
3e-0
123456789.123456789
1.0000000000000000
0.1000000000000000
```

Nicholas LaJoie, ECE 331, HW 3

```
0e123
123e0
1E3
-2E3
-4E-5
6.E-6

Numbers that shouldn’t match:
1
100000
000001
0
18
e3
E3
-E3
-e3
+E3
+e3
+1
-1

Hi Sheaff!

Some more tests:
This isn’t a floating point number: 01
But this is: 0.1
1e4
.0
0.
-0.
+0.
+0e0
0.0
1.00
0123
1.23456
12345.6789
Another sentence to make sure it works.
This sentence has a floating point in it: --> 3.14 <-- Wow!
Alright, almost done tests.
-0
0-
1.149-
+83.5
+0
0+
When will this end??
Line number: 42
The final number: +3.14159
Pi time!
```

5. Number of packages command: `apt-cache pkgnames | egrep [0-9] | wc`

6. Shortest ERE for 0-100: `^(100|[0-9]|[1-9][0-9])$`

7. ERE for -Wselector - in the gcc man page, search:
   `[[:space:]]{3,}-Wselector`