LAB 9 — UNIX Shell Scripting

Problem A

1. Specification

Create a Bourne shell script file called **`phone`** that will search a file in your current working directory called **`phone_book`** for telephone numbers and names and print out each line that matches the pattern entered.

2. Implementation

- The script should prompt the user for a name as follows:

```
% phone
Enter the name to search: alex
alex johnson    (416) 555-1234        family doctor
Alexander  Smith (905) 555-9876        home renovation contractor
```

- The pattern to be searched for is case-insensitive. That is, the entries displayed include substrings `alex`, `ALEX`, `Alex`, etc.

Problem B

1. Specification

As Problem A.

2. Implementation

As Problem B, except that the name to be searched for is now entered as a command-line argument:

```
% phone alex
alex johnson    (416) 555-1234        family doctor
Alexander  Smith (905) 555-9876        home renovation contractor
```

Problem C

To display a file on/as a web page, it must be readable by all: `chmod a+r file_name`

A directory must be executable and readable by all: `chmod a+rx dir_name`

Write a script called `mkpub` (make public) that takes a directory or file name as a command line argument. It then sets the appropriate permission(s) for the directory or the file, and displays a confirmation message. If the file/directory does not exist then display an error message as in the example shown below.

Following are a few examples:

```
% mkpub Temp
Directory 'Temp' is now made public.

% ls -ld
drwxr-xr-x  2 bil faculty 4096 Nov 20 18:20 Temp/

% mkpub Temp/example.c
File 'Temp/example.c' is now made public.

% ls -l Temp/example.c
-rw-r--r--  1 bil faculty 0 Nov 20 18:19 Temp/example.c

% mkpub ghost.txt
File 'ghost.txt' does not exist.
```

Problem D

As problem C, except that the user may now enter more than one command-line argument. Following is an example.

```
% mkpub Temp Temp/example.c ghost.txt
Directory 'Temp' is now made public.
File 'Temp/example.c' is now made public.
File 'ghost.txt' does not exist.
```

Problem E

UNIX command `rmdir` only removes empty directories. Write a script called `myrmdir` that removes non-empty directories. It takes one command line argument, which is the directory to be removed. If the directory does not exist or the argument is a file name, display an error message "`Not a valid directory`"

Program first removes the files inside the directory. For each file, display a prompt (as shown below) asking the user if he/she wishes to remove the file (to prevent accidental deletions). If the response is "`y`" or "`Y`", the file is removed. For any other responses, the file is not removed and a message is displayed to confirm that the file has not been removed.

Finally remove the directory with `rmdir`. If the directory can be removed by `rmdir` then display a confirmation message as follows.

```
% myrmdir Temp
```

```
remove file 'Temp/README'? y
remove file 'Temp/ex1.c'? y
remove file 'Temp/lab1.out'? y
remove file 'Temp/lab2.out'? y
remove file 'Temp/lab3.out'? y

% ls -ld Temp
ls: cannot access Temp: No such file or directory
```

Note that the above error message is displayed by the UNIX system command `ls` itself, to show that directory `Temp` was successfully removed.

Following is another example:

```
% myrmdir Temp
remove file 'Temp/README'? y
remove file 'Temp/example.c'? n
File 'Temp/example.c' not removed.
remove file 'Temp/lab1.out'? y
remove file 'Temp/lab2.out'? y
remove file 'Temp/lab3.out'? y
rmdir: `Temp': Directory not empty
```

Note that the above `rmdir` error message is from the UNIX system command `rmdir` itself. Thus you do not have to display your own message if the directory to be removed is not empty.

Problem F

Implement a simple calculator using UNIX shell scripting that accepts input in the following format and displays the result of the computation:

`calc [operand_1] [operator] [operand_2]`

The operands `operand_1` and `operand_2` are integers. The operator is one of the following: addition (+), subtraction (-), multiplication (x), division (/) and modulo (%).

Note: For the multiplication operator in the command line arguments, use letter **`'x'`**. If you use the asterisk **`'*'`**, your program will not work properly.

Hint: Use the `expr` utility. The arithmetic operators are the same as those in C or Java. However, the multiplication operator **`'*'`** requires the use of a backslash in the code, e.g., `$int1 \* $int2`.

Following are a few examples:

```
% calc 200 + 100
300
% calc 100 - 450
-350
```

```
% calc -50 x 50
-2500
% calc 30 / 7
4
% calc 29 / 12
2
```

Assume that all command line arguments are valid, and no error checking is required.