UNIX Shell Scripts

The UNIX assignment consists of two parts.

Part 1

Part 1 is worth 45% of assignment 2. Problems 1 to 6 are 5% each, and problem 7 is 15%.

Problem 1

Write a script named **isearch** to allow users to input a file name and a word/pattern. Search the file for the word/pattern and display the lines that contain the word/pattern on the standard output. If the word/pattern is not found, display a message.

```
indigo 20 % isearch
Enter the file name: phone_book.txt
Enter the word to search for: from
Case-sensitive (y/n)? n
Nikola Sandic   (647) 209-7932 friend from work
ROB Oxley       (416) 723-8759 friend from university
```

If the word/pattern is not found, display a message.

```
Word ‘from’ not found.
```

Problem 2

Write a script named **icount** to allow users to input a file name and display on the standard output the number of lines, words or characters in the file.

```
indigo 21 % icount
Enter the file name: phone_book.txt
Count lines, words, characters or all three (l, m, c, a)? b
Invalid option
Count lines, words, characters or all three (l, m, c, a)? m
File ‘phone_book.txt’ contains 125 words.
```

3. Write a script named **allfiles** to allow a user to list all (ordinary) files in the current directory, one per line, but not the sub-directories in the current directory. (*Note*: command ‘`ls`’ displays all files and directories.)

```
indigo 22 % allfiles
lab3a.c
lab3b.c
lab3b_out.txt
```

4. Write a script named **fdisplay** to allow a user to input a file name and an option to display the content of the file (**e**ntire file, entire file one **p**age at a time, **f**irst 10 lines, **l**ast 10 lines). Display the content of the file as the user specifies. If the user enters an invalid option, display an error message and ask for a correct option.

```
indigo 23 % fdisplay
Enter the file name: phone_book.txt
Enter option (e, p, f, l): a
Invalid option
Enter option (e, p, f, l): p
# File is displayed one page at a time
```

5. Write a script named **`doublesp`** to allow a user to input an input and an output file name. Copy the content from the input file to the output file, adding a blank line between every two lines of the input file (double spacing). If the output file exists, display the message shown below. If the user answers yes then append the output to the file. If the user answers no, exit and do nothing (i.e., do not overwrite an existing file).

```
indigo 24 % doublesp
Enter input file name: phone_book.txt
Enter output file name: outfile.txt
File 'outfile.txt' exists.
Append to it (y/n)? y
```

6. Write a script named **`addlines`** to allow a user to enter an input and an output file name. Copy the content from the input file to the output file, line by line, adding a line number at the beginning of each copied line. If the output file exists, display a message as above. If the user answers yes then append the output to the file. If the user answers no, exit and do nothing.

```
indigo 25 % addlines
Enter input file name: phone_book.txt
Enter output file name: outfile.txt

indigo 26 % cat outfile.txt
1: Ray Szeto       (416) 564-6786  friend
2: alex johnson    (416) 555-1234  family doctor
3: Tom Podstawka   (647) 579-3854  dentist
…
```

7. Write a script named **`prtlines`** to allow a user to input a file name and two numbers **x** and **y**. First show the number of lines in the input file. Then display line **x** to line **y** on the standard output, adding a line number at the beginning of each displayed line. Check the validity of the file name and the two numbers.

```
indigo 27 % prtlines
Enter the file name: phone_book.txt
File 'phone_book.txt' has 23 lines.
From line: 3
To line: 5
3: Tom Podstawka   (647) 579-3854  dentist
```

```
4: Greg Ivan       (905) 789-6703  neighbour
5: Nikola Sandic   (647) 209-7932  friend from work
```

If there is an error with the input line number, display an error message and prompt for the correct line number as shown below.

```
From line: -3
Invalid line number
From line: 30
Invalid line number
From line: 3
To line: 2
Invalid line number
To line: 5
```

Notes

- An input file must exist and be readable before being read from.
- An existing output file must be writable before being written to.
- Assume that all inputs from the user are case-insensitive, except file names. (Unix file names are case-sensitive, so we keep the same convention.) For example, the user may enter either 'y' or 'Y'.
- Assume that all input files are text files containing no backlash characters.
- In problems 6 and 7, add a space after the colon before writing the line.

Sample Error Messages

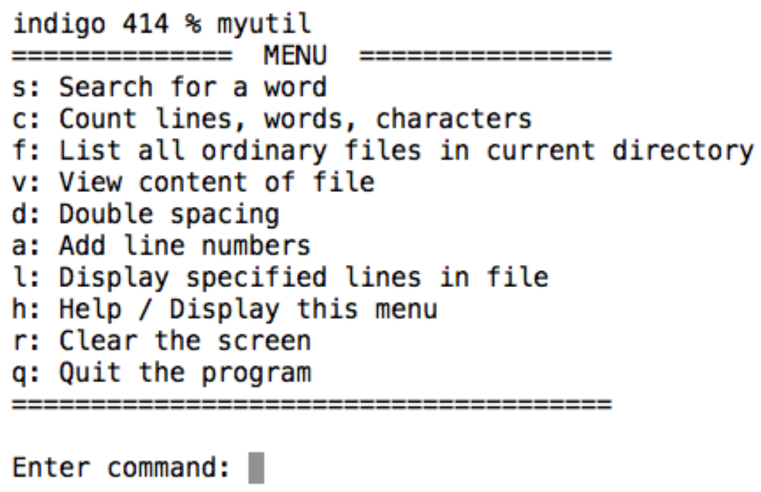
```
Enter input file name: yourfile.txt
File 'yourfile.txt' does not exist.

Enter input file name: ourfile.txt
File 'ourfile.txt' is not readable.

Enter output file name: hisfile.txt
File 'hisfile.txt' is not writable.

Enter output file name: outfile.txt
File 'outfile.txt' exists.
Append to it (y/n)? n
```

In the above examples, simply exit the script upon the file error or an answer 'n'.

Part 2

Part 2 is worth 25% of assignment 2. Part 2 uses all the scripts from Part 1 to form a small Unix program.

Write a script named **`myutil`**. When the script is first executed, the following menu is displayed, followed by a prompt "`Enter command:` "



```
indigo 414 % myutil
==============  MENU  ================
s: Search for a word
c: Count lines, words, characters
f: List all ordinary files in current directory
v: View content of file
d: Double spacing
a: Add line numbers
l: Display specified lines in file
h: Help / Display this menu
r: Clear the screen
q: Quit the program
======================================

Enter command: 
```

The user then enters one of the commands displayed on the menu. Based on the user's command, `myutil` will "call" one of the scripts in Part 1 or perform one of the following tasks:

- s: Execute the script **`isearch`** above to search a file for a word/pattern and display the lines that contain the word/pattern on the standard output.
- c: Execute the script **`icount`** to display on the standard output the number of lines, words and/or characters in a file with one of the specified options (`l, m, c, a`).
- f: Execute the script **`allfiles`** above to list all ordinary files in the current directory, one per line, but not the sub-directories.
- v: Execute the script **`fdisplay`** above to view the content of a file with one of the specified options (`e, p, f, l`).
- d: Execute the script **`doublesp`** above to copy a text file to another and add double line spacing.
- a: Execute the script **`addlines`** above to copy a text file to another and add a line number to each copied line.

- l: Execute the script **`prtlines`** above to display on the standard output a subset of lines in a text file.

- h: Display the menu. (After many commands were executed, the menu would be pushed out of view.) After displaying the menu, prompt for the next command.

- r: Clear the screen (standard output) using the Unix command "`clear`" and prompt for the next command.

- q: Quit/exit the script `myutil`.

After a command is executed, the user is prompted for the next command, until command "**`q`**" is entered.

If one of the seven scripts from Part 1 exits with an error (e.g., invalid user input, unreadable file), continue the script `myutil` by prompting for the next command. The user can repeat the same command and then enter a valid input or file name this time.

Add a blank line before the prompt "`Enter command:` " to make the screen more readable.

If a command is not valid, issue an error message "`Invalid command`" and prompt for the next command.

Assume that all commands from the user are case-insensitive. For example, the user may enter either 'q' or 'Q' to quit.

Do not merge all scripts from Part 1 into a big file. Simply leave them in the same directory as the script `myutil` and `myutil` will find the appropriate file when executing the corresponding script/command. You may create additional scripts to be "called" by `myutil`, e.g., creating a script to display the menu.