CS2211a Assignment 2

Issued on: Thursday, October 3, 2013

Due by: 11:55 p.m. on Thursday, October 10, 2013

- For this assignment, ***only electronic submission*** at owl.uwo.ca is required.
- ONLY user ***Courier New*** font.
- Leave ***one empty line*** and ***a line of "$"*** between the answer of each question.
- ***Write the question number is a separate line followed by an empty line***
- *When you cut-and-paste any Unix command or Unix output, please make them* ***BOLD***

Your submission should look like that:

```
Q1

Write here your answer. ls /faculty/elsakka Write here your answer.
Write here your answer.
Write here your answer.

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Q2

Write here your answer.
Write here your answer.
Write here your answer.

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

and so on

Failure to follow the above format may cost you 10% of the total assignment mark.

- Late assignments are strongly discouraged
  - 10% will be deducted from a late assignment (up to 24 hours after the due date/time)
  - After 24 hours from the due date/time, late assignments will receive a zero grade.

QUESTION 1 (15 marks)
Write a Bourne shell script **lastarg**, which takes 0 or more arguments and prints the last (rightmost) argument in the argument list. ***You can assume that arguments will be made up of letters and digits only.***

Note that, the number of arguments can be more than 10. For example, the output of

```
lastarg arg1 arg2 arg3 arg4 arg5 arg6 arg7 arg8 arg9 arg10 arg11
```

should be

```
arg11
```

If no argument is provided, simply return nothing.

You should write enough inline comments inside your shell script to explain each part of it.
In your report, you should include test cases to demonstrate all possible options in your program.

If **lastarg** is placed in your home directory, what will happen if you execute the following command? Explain why you got this output.

```
cd; lastarg .*
```

QUESTION 2 (15 marks)
Write a Bourne shell script **odd_prn**, which echoes ***its shell script file name*** as well as ***the values of its odd arguments***. Even arguments should be ignored. Each value should be echoed in a separate line. ***You can assume that arguments will be made up of letters and digits only.***

Note that, the number of arguments can be more than 10. For example, the output of:

```
odd_prn to C or not to C that is the question
```

should be

```
odd_prn
to
or
to
that
the
```

If no argument is provided, simply return the shell script file name only.

You should write enough inline comments inside your shell script to explain each part of it.
In your report, you should include test cases to demonstrate all possible cases in your program.

If **odd_prn** is placed in your home directory, what will happen if you execute the following command? Explain why you got this output.

```
cd; odd_prn .*
```

QUESTION 3 (35 marks)

Write a ***~~flow chart~~ pseudo code*** and a ***Bourne shell script*** that causes the following output (below) to be displayed.

Note that, there is a single space between each value.

The number of column should be taken as input.

For example, if the input to the program is **6**, the program should produce the following output:

```
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
0 1 2 3 4 5
0 1 2 3 4
0 1 2 3
0 1 2
0 1
0
```

You should write enough inline comments inside your shell script to explain each part of it.

QUESTION 4 (35 marks)

Write a ***~~flow chart~~ pseudo code*** and a ***Bourne shell script*** (called **nums**), which takes two arguments and gives the following output:

Command	Result (output sent to stdout)
nums 0 input-file	The smallest and the 2nd smallest numbers inside **input-file**
nums 1 input-file	The largest and the 2nd largest numbers inside **input-file**

Assume that **input-file** includes a list of ***unsorted*** numbers (one number per line). *Your program should work with any* **input-file** *that includes numeric values (positive, negative, and/or zero), one value per line.*

If the number of arguments is zero, one, or more than two, the script should produce a message saying "**Usage: nums option input-file**" and exit with status value = 1 (i.e., *unsuccessful*).

If the file does not exist, the script should produce a message saying, "**input-file not found**" and exit with status value = 2 (i.e., *unsuccessful*), where **input-file** is the *provided file name (i.e., the second argument).* If the first argument is neither 0 nor 1, the script should produce a message saying, "**Option must be 0 or 1**" and exit with status value = 3 (i.e., *unsuccessful*).

When the program is *successful* (i.e., does the required job), the exit status should be 0.

You can test the exist status of the program by executing "**echo $?**" *immediately after running the program*.

You should write enough inline comments inside your shell script to explain each part of it.

Create a file called **numbersfile**. Write the following numbers inside that file, one number per each line, (**3, 6, 9, 11, 3, 4, -8, -10, 0, 16, 5**). Test your script using at least the following cases (include the output in your report):

```
nums  ; echo $?
nums 0; echo $?
nums 5; echo $?
nums 0 numbersfile; echo $?
nums 1 numbersfile; echo $?
```

```
nums   numbersfile; echo $?
nums 5 numbersfile; echo $?
nums 0 numbersfile aaaa; echo $?
nums 0 aaaa; echo $?
nums 1 bbbb; echo $?
```

tka.