# Shell script IV.

## Login into the server of the subject (Linux) and create a subdirectory!

## Regular expressions, grep, sed, awk

**Remember some rule about regular expressions:**
**[a-z]** – means the character set between 'a' and 'z'
**^ -** means it must fits to the first position, **$ -** means the same with the last position
**+ -** means that it iterates once or more, *** -** means the iteration can be 0 as well
**( )** – creates a group
**\ -** can change the default meaning of a character
**|** - alternative
**There are a lot of rules!**
**See them e.g. at http://www.regular-expressions.info/tutorial.html**

a) Write a script which gets a filename given by a parameter and writes out on the screen only the lines which contain numbers! (grep)
b) Modify the above given script and write out only that lines which contain only numbers! (grep)

**Remember some options of grep (the others are in man):**
**-v** the reverse regular expression
-c count the number of fitting lines to pattern

c) Write a script which gets a filename as a parameter and counts how many lines are in it which does not contain numbers! (grep, wc –or grep -c)
d) Write a script which gets a filename and a word as parameters. The program has to copy only that lines into a new file which does not contain tha given word! (grep -v)
e) Write a script which gets a filename as a parameter and writes out only that lines which does not contain the word *apple* or *pearl*! For the solution use grep twice in a pipe! (grep -v)

**Remember the usage of sed :**
sed 's/exp1/exp2/' - change exp1 to exp2
sed 's/exp1/exp2/g' - change exp1 to exp2 in each ocasion
sed '/exp/d' – delete exp

f) Create a second version of the above mentioned script using sed! (sed //d)

**Remember some useful details of awk:**
awk 'BEGIN {executed at start} { executed for each lines } END {once at the end}'
$1, $2 … the words in a line – separated by a space
awk ' value ~ pattern {} ' - executes only for the fitting lines (!~ does not fit)

g) Create a third version of the script using awk!

h) Write a script which change a given word to an other one (each of them). The filename and the words are given by parameters. (cat, sed s///g)
i) In a file there are numbers (one number per line) between 10 and 99! Create a script which writes out the numbers in a reverse order. (E.g instead of 15 give 51). The filename is given by a parameter! (sed s///)
j) In a file there are two words per line – the words are separated by a space. Write a script which changes the order of words in a line! Use sed for the solution! (sed s)
k) Write a new version of the solution – use awk for it! (awk)
l) In a file there are numbers only one line by line. Write a script which adds them all! Use file reading and input redirection for the solution. The filename is given by a parameter. (while read…< file)
m) Create a new version of the task with awk! Use BEGIN for giving a starting value, and END part to print out the result! (awk)
n) In a file there are numbers and words mixed. Copy that lines which contain not only numbers into  a new file! The filename is given by aparameter. (grep -v)
o) Implement the same task with awk too!
p) Write a script which reads a number from the keyboard and write it out vice versa! (E.g. 123 as 321) Scatch: while [ $number -gt 0] do $actual=$(( $number % 10 )); $number=$(( $number / 10 )); reverse=$( echo ${reverse}${actual}); done)
q) Write a script which reads in a number from the keyboard and adds the digits of it! E.g. 123 gives 6 (1+2+3)
r) Write a script which writes out the lines of a file in reverse order! The first line last etc. Use an array for storing the data temporarily!

## Finish your work and logout from everywhere!