Shell script II.

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

Review

Arithmetical operations

Remember: Be careful to use spaces inside the expressions! Use expr command to evaluate the expression! Do not forget about the command substitution ` `!
Before *, use \ !

a) Why will you get an error message if you type the following command?
   a=expr 2 + 3 (command substitution)
b) Why will you get an error message if you type the following command?
   a=`expr 2+3` (missing spaces)
c) Why will you get an error message if you type the following command?
   a= `expr 2 + 3` (must not write a space just after the =)
d) Why will you get an error message if you type the following command?
   a=`2+3` (missing expr command)
e) Write a script which adds 1 to the content of a file (there is only a single number in the file) and writes out the result back into the file! The filename is given by a parameter! (cat, ``, expr, >,$1)
f) Write a script, which calculate the quadrat (square) of the number given in a file! (cat, ``,$1)

Input, read

a) Write a script which can copy a file with a nice „user interface”! Read in the name of the source file and the destination! (read, cp)
b) Write a script which reads in a world and then writes it out nicely, in big letters! (read, echo word | figlet)
c) Write a script which reads in two numbers and calculate their distinction (a-b)! (read, expr)

Grep, searching patterns

a) Write a script which searches for files in a directory and in it's subdirectories (recursively) which contain a given word! The directory and the word are given by parameters. (grep –lr $1 $2)
b) Write a script which lists out that lines of a file which contain a given word. The filename and the word is given by parametrs. (cat $1| grep –w $2)
c) Write a script which writes out that lines of a file which are ended by a given word! The filename and the word is given by parametrs. (cat $1| grep –w 2)

Tr, change characters

a) Write a script which changes the small characters of a word to capital letters! The word is given by a parameter. (echo, tr [a-z] [A-Z])
b) Write a script which changes the () brackets to {} brackets in a file! The word and the filename are given by parameters. (cat, tr)

c) Write a script which changes the () brackets to {} brackets and the original {} brackets to [] brackets in a file! The word and the filename are given by parameter. Think of the order of the commands! (cat, tr)

Program structures

Branches:

1) test command (gives back a logical value – 0 or 1)

Remember the usage! Do not miss the required spaces!
test operand1 operator operand2 *or* [operand1 operator operand2]

Operators at numbers:	-lt, -gt, -le, -ge, -eq, -ne
at strings:	=, !=
at files:	-f, -d
Compound expressions:	-a, -o, !

a) Firstly try test command using simply the command line!
E.g. test 5 –lt 6
echo $?
Examine the result! Try it with test 5 –lt 2 as well!

b) Decide whether a given number is greater than 10 or not! The number is given by a parameter. (test $1 –gt 10; echo $?) $? *Contains the result of the evaluation!*

c) Decide whether a given word is equal to your nick name or not. The word is given by a parameter. (test „$1” = „your nick name”; echo $?)

d) Decide whether a given word is not equal with „ELTE” or not. The word is given by a parameter. (test, echo, $?)

e) Decide whether a given number is between 0 and 5. The number is given by a parameter. (test, echo, $?)

f) Decide whether a given number is between two numbers N and M. The number and the ends of the intervall are given by parameters. (test, $?)

2) if instruction

Remember the usage! Remember to write every instruction into a new line!

```
if test logical expression
        instructions
then
        instructions
fi
```

```
if test logical expression
        instructions
then
        instructions
else
        instructions
fi
```

```
if test logical expression
        instructions
then
        instructions
elif test logical expression
        then
                instructions
fi
```

a) Write a script which reads in a number and decide whether it is equal to, smaller or greater than 10! Write out the result to the screen! (read, test, if, echo)

b) Write a script which reads in a number and decide whether it is divisible by 10 or not! Write out some text („divisible”/”not divisible”) due to the result on the screen! (read, test, if, echo)
c) Write a script which greets you properly („Good morning” before noon, „good afternoon” during the day and „Good evening” in the evening.)! (date, if, test, echo)
d) Create a script which calculates the square (quadrat) of the value given in the value.txt file and modify the file content with the result! Check whether the file is exist or not! (expr, cat,``,>, if)
e) Modify your script which adds two number parameters with checking the existence of the parameters! ($#,if, test,expr,``)
f) Modify your .profile file to greet you properly due to the login time! (date, test, if, echo)
g) Modify your .profile file to greet you on your birthday! (date, test, if, echo)

3) case instruction

Remember the usage! Do not miss the required spaces and new lines!
```
case pattern in
  pattern1)
     instructions
     ;;
  pattern2)
     instructions
     ;;
  …….
  *)
     instructions to be executed otherwise
     ;;
esac
```

a) Write a script which executes different commands due to it’s parameter. If the value of the parameter is –l then list the actual directory, if it is –d then write out the date, if it is –w then write out the names of the logged in users! (case, ls, who, date,$1)
b) Modify the above created script and write out a help information (the accepted parameter values) if the user wants to execute it without or with wrong parameters! ($1, case, who, ls, date, echo)
c) Can you write the script without using case instruction? If you can, write it please – if not, explain why not!
d) Write a script which can decide wheteher it’s number parameter is inside an interval or not. Th script should be used with two or three parameters. In each cases the first parameter is the examined number. If you use it with 2 parameters, the second parameter will mean the higher end of the interval (the lower end will be 0) otherwise the second and third parameters are the ends of the interval. ($#,$1,$2,if, test, echo)

Cycles:

1) For cycle

Remember the syntax! Do not miss the required spaces and new lines! Use seq command to create a list of values – if you want to execute something a given times!

```
for var in value1 value2 ... valueN
do
  instructions to be executed for every vale
done
```

a) How many times will be written out „Hello” by the undergiven script? Explain the result!

```
for i in seq 5
do
        echo Hello
done
```

b) Write a script which writes out 5 times a tongue-twister: „*If two witches were watching two watches, which witch would watch which watch?*”! (for, echo, seq)
c) Modify the above created script with a parameter, which gives how many times you should write out the tongue-twister: „*If two witches were watching two watches, which witch would watch which watch?*”! (for, echo, seq, $1)
d) Write a script which greets with „Hello” each of the logged in users by their login name! You have to greet everybody only once – so you have to be careful with users who are logged in twice or more! (for, who, cut ,sort, uniq)

2) while cycle

Remember the syntax! Do not miss the required spaces and new lines!

```
while logical expression
do
   instructions to be executed if logical expression is true
done
```

a) Write a script which writes out 7 times a tongue-twister: „*If you want to buy, buy, if you don't want to buy, bye bye*”! This time use while cycle for implementation the task! (while, echo)
b) Write a script which reads in 5 numbers and adds them. (read, while, echo, expr, ``)
c) Write a script which reads in words till it gets the word „end”. (while, read)

3) until cycle

```
Remember the syntax! Do not miss the required spaces and new lines!
until logical expression
do
        instructions to be executed until logical expression is true
done
```

a) Write a script which writes out 10 times a tongue-twister: „*Give papa a cup of proper coffee in a copper coffee cup*”! This time use until cycle for implementation the task! (until, echo)

b) Write a simple menu system with the help of until cycle! With typing 1, the program should execute a script which adds two numbers. With typeing 2, the program should execute a script which multiply two numbers. At 3, you have to exit your program – otherwise it should continue it’s working by waiting the required nembers! (until, read, scripts, case)
The scatch of it:

```
until
 clear
 echo Choose a program!
 echo 1 Sum
 echo 2 Product
 echo 3 Exit
 echo -n Type int he number:
 read number
 [ $number -eq 3 ]
do
  case $number in
   1)  echo Summing
       ;;
   2)  echo Producting.
       ;;
   *)  echo Not good!
       ;;
  esac
  sleep 5
 done
echo Happy to have been chosen by you!
```

Finish your work and logout from everywhere!