## A) Examples of shell variables.

**Create your own shell variables by setting content to them, for instance PHONENR, MYNAME etc. How can you check that they are not empty?**

```
:~$ PHONENR='11223344'
:~$ MYNAME='ik0v'
:~$ echo $PHONENR
11223344
:~$ echo $MYNAME
ik0v
```

**Change content in variable PHONENR. Check if it worked.**

```
:~$ PHONENR='0047'$PHONENR
:~$ echo $PHONENR
004711223344
```

**Set variable PHONENR to be empty. Check result.**

```
:~$ PHONENR=
:~$ echo $PHONENR          - no value
```

**List out all variables in shell with set command. Did you find your variables?**

```
:~$ (set -o posix ; set) | less

:~$ set | grep MYNAME
MYNAME=ik0v

:~$ set | grep PHONENR
PHONENR=
```

## B) Environmental variable PATH

**Start a new shell from terminal with bash command. In this new shell run command PATH= , or equal to nothing. What did you do with that command?**

:~$ bash
:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin
:~$ PATH=
:~$ echo $PATH - no value

At a first I started e new shell. Then I checked PATH variable before changes. Looks like standard value was inherited from original shell. Then PATH variable was set to point to nothing, and next line confirms it.

**Try ls command. Does it work now? Explain.**

:~$ ls
bash: ls: No such file or directory

ls command doesn't work because path to that command was deleted. In other words, new shell doesn't know where to look for that command.

**Go back to previous shell with exit command. Does ls command work now? Check PATH variable. Explain.**

:~$ exit
:~$ ls - ls works now, content of home directory was listed out.

:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin

PATH-variable was changed in new shell, but not in first one. Now we came back to that and PATH variable wasn't changed. Therefore, ls and other command works fine here.

**Try this: MYPATH=$PATH. What happens here?**

:~$ MYPATH=$PATH

Here we created a new variable called MYPATH. At the same time MYPATH was assign same value as PATH environmental variable.

:~$ echo $MINPATH - /usr/local/bin:/usr/bin:/bin

**Try to extend PATH with folder games from your home directory. What did u do?**

:~$ PATH=$PATH":/usr/games"
:~$ echo $PATH - /usr/local/bin:/usr/bin:/bin:/usr/games

First line updated PATH variable and added a new folder, /usr/games. Different folders are separated by semicolon.

## C) Aliases

**Check which aliases are set up in your profile. Create some new aliases, for example alias lh to list out files in "human-readable" format and lr to list out files in reverse order comparing to standard ls command, or oldest files first. In both cases use option -l.**

:~$ alias
…. here comes a list with some aliases, 4 in my case.

:~$ alias lh="ls -lh"
:~$ alias lr="ls -lr"

:~$ lh
:~$ lr
Both aliases works fine.

**Create an alias to list out all pdf files from your system. Send errors to null device.**

:/$ alias pdfs='find / -type f -name "*.pdf" 2> /dev/null'

:/$ type pdfs
pdfs is aliased to 'find / -type f -name "*.pdf" 2> /dev/null'

:/$ pdfs - about 15 pdf files were found, got no error messages.

**Where would you add alias in order to have a permanent effect from it? Use grep command and check if you can find existing aliases in certain files.**

:~$ touch .bash_aliases
In this files aliases are saved permanently.

:~$ less .bashrc | grep alias
Here comes a list with existing aliases.

## D) .inputrc

**Create a file ~/.inputrc with the following contents:**
**"\e[A": history-search-backward**
**"\e[B": history-search-forward**

:~$ touch .inputrc
:~$ nano .inputrc - adding both lines, saving file

**Restart bash, and type one or two characters of a command you have executed before. Then press the Up arrow, you'll see previous commands that match.**

That works fine, it's a helpful feature.