TD 5 - ISS

Exercice 1.

1.1

```
A: val_de_a
B: val_de_b
C: val_de_b
B: val_de_b
A: val_de_a
C:
A: val_de_a
```

Cet affichage est dû au fait que dans le cas de A, `var` n'est pas une variable d'environnement du processus (à la différence du script `B.sh`) et donc n'est pas herité lors du `fork()` et `exec()`.

1.2

C'est parce que à chaque nouveau shell tout est perdue, pour sauvegarder une telle modification de façon permanente faut modifier le fichier de configuration du shell.

1.3

Oui.

1.4

On n'a pas besoin de l'écrire parce que `source` cherche d'abord dans le répertoire courant.

```
A: val_de_a
B: val_de_b
C: val_de_b
B: val_de_b
A: val_de_b
C: val_de_b
A: val_de_b
```

1.5

Exercice 2.

2.1

```
#! /bin/bash
# majuscule.sh
while IFS= read -r line; do
    echo "${line^^}"
done
```

2.2

```
#! /bin/bash
# numerotation.sh
idx=0
while IFS= read -r line; do
    echo "Ligne ${idx} : ${line}"
    idx=$((${idx} + 1))
done
```

2.3

```
#! /bin/bash
# MajusculeEtNumerotation.sh
idx=0
while IFS= read -r line; do
    echo "Ligne ${idx} : ${line^^}" >> ./temp.txt
    idx=$((${idx} + 1))
done
```

2.4 et 2.5

À FAIRE

Exercice 3.

3.1

```
$ head -n 1 notes.lst
```

3.2

```
$ grep -E "Satomi" notes.lst | cut -d ' ' -f 5-7
17 5 3
```

3.3

```
$ tail -n 1 notes.lst | cut -d ' ' -f 5-7
```

3.4

```
$ head -n 2 notes.lst | tail -n +2 | cut -d ' ' -f 5-7
```

3.5

```
$ tail -n +2 notes.lst | cut -d ' ' -f 1 | uniq | wc -l
```

3.6

```
$ f=$(tail -n +2 notes.lst | grep "F" | wc -l)
$ h=$(tail -n +2 notes.lst | grep "H" | wc -l)
$ if [[ $f -eq $h ]]; then echo "Parité respecté!" else echo "Nein" fi
```

3.7

```
$ satomi=$(grep "Satomi" notes.lst | cut -d " " -f 5-7)
$ for x in $satomi; do
$     res=$((${res} + $x))
$ done
```

3.8

```
$ notes=$(tail -n +2 | cut -d " " -f 5)
$ #loop
```

3.9

à faire

Summary

Bash behavior

- Uppercase of a variable: `${var^^}`
- Lowercase of a variable: `${var,,}`

Commands

`wc`

Displays, in this order, the number of lines, words, characters, and/or bytes in a file.

- `-l` : Number of lines
- `-w` : Number of words
- `-m` : Number of characters
- `-c` : Number of bytes

`head`

Displays the first `x` lines of a file.

- `-n x` : Number of lines to display
- `-n -x` : Displays all without the last `x` lines

`tail`

Displays the last `x` lines of a file.

- `-n x` : Number of lines to display
- `-n +x` : Displays all lines starting from the `x` th line

`sort`

Sorts lines of text files.

- `-n` : Sorts numerically
- `-r` : Reverses the sort order (descending)

`uniq`

Removes duplicate lines from a sorted file. Note that the input must be sorted for `uniq` to work correctly.

`grep`

Searches for lines matching a given pattern.

- `motif` : The pattern to search for
- `-v` : Inverts the match, displaying lines that do not match the pattern

`cut`

Extracts sections from each line of a file.

- `-c` : Selects characters based on their positions. For example, `cut -c 2,4-6` selects the 2nd, 4th, 5th, and 6th characters.
- `-d` : Specifies the delimiter for field separation (default is tab).
- `-f` : Selects fields based on the delimiter. For example, `cut -d ' ' -f 2,4-6` selects the 2nd, 4th, 5th, and 6th fields.