1st Laboratory Exercise Software Development Tools and Systems Programming

Eleftheriadis Dimitris 2015030067

Regr program

The first issue of the laboratory exercise was the implementation of the regr program in Bash shell, which will accept as input a list of files. The lines of the files are in the format num1:num2. We were given the functions with which we will calculate the parameters of the linear regression and we implemented the program as follows: Initially, after taking the parameters from the respective input file, we created a temporary file and saved the two fields without spaces with the following command

```
awk 'NF' $1 > "${DIR}temp.txt"
```

We use the “ > ” operator which creates the temp.txt file if it does not exist, otherwise it overwrites it.

Calculation of Sums

- To calculate the length (lines of the file) lines of the file) the following command was used:

  ```
  cat ${inputFile} | wc -l
  ```

- Then, the sum of the left column of the file was calculated using the following:

  ```
  var_x=$1 awk -F: '{x_array+=$1}END{print x_array}' ${inputFile};
  ```

  which, after reading the input file, divides its data into fields left and right of the colon (lines of the file) and then adds them together and finally saves the final result into variable var_x.

- The calculation of the remaining sums is based on the same logic.

- Note that all the above procedures were implemented in functions.

- The calculation of parameters a and b was carried out with the help of the above sums. For the operations we used the command bc -l, because the bash shell can not do the calculation with decimal digits.

➢ To implement the error we did a while building the final result.

Checks

➢ The program checks if the respective file has less than three lines.

Issue 2. Results program

For this program we were asked to calculate the scores, as well as the total goals for and against each team and at the end to display them aligned. The matches are given as entries from a file.

File reformat

➢ First, we remove all the tabs and blank lines that the input file may contain and save it in a temporary file.

➢ In the next step, we created two tables for the 2 fields (lines of the file) rival teams in each match) with their score and their score, using the commands

```
mapfile -t pointsHome < <(file lines) awk -F'[:-]' '$3 > $4 {printf "%s:3:%d:
%d\n",$1,$3,$4;} $3 < $4 {printf "%s:0:%d:%d\n",$1,$3,$4;} $3 ==  $4{printf
"%s:1:%d:%d\n",$1,$3,$4;}' ${inputFile} )
mapfile -t pointsAway < <(file lines) awk -F'[:-]' '$3 > $4 {printf "%s:0:%d: %d\n",$2,$4,$3;}
$3 < $4 {printf "%s:3:%d:%d\n",$2,$4,$3;} $3 ==  $4{printf "%s:1:%d:%d\n",$2,$4,$3;}'
${inputFile} ).
```

We then assembled the two tables into one, ready for editing.

Calculations and subtraction of duplicates

➢ The table was then saved in a file in the form: team name with its score and the total goals it has scored and the total goals it has conceded. With this file, we calculated and saved in tables, for each group its final data (files of the file) name, total points, total goals for and against), deleting the duplicates. The commands that do this are:

```
mapfile -t finalStand < <(file lines) awk -F'[:-]' '{ seen[$1]+=$2 } END  { for (file lines) i
in seen) printf " %s: %4d\n", i, seen[i] } ' "$ {DIR}temp.txt" )

mapfile -t totalGoalsFor < <(file lines) awk -F'[:-]' '{ seen[$1]+=$3 } END { for (file lines) i in
seen) printf " %d\n", seen[i] } ' "$
{DIR}temp.txt" )

mapfile -t totalGoalsAgainst < <(file lines) awk -F'[:-]' '{ seen[$1]+=$4 }  END { for (file
lines) i in seen) printf "%d\n", seen[i] } ' "$
{DIR}temp.txt" )
```

➢ Then, in while loop we build the formatting of the final file with the

command

```
myTemp1[$s]=$(file lines) echo "${finalStand[s]} ${totalGoalsFor[s]}-$
{totalGoalsAgainst[s]}")
```

Sorting and printing

➢ Then we sort our file based on the scores and in case of a tie, based on name, with the command

```
mapfile -t finalStandings < <(file lines) sort -r -n -t':' -k 2n -k 1n "$ {DIR}temp1.txt" )
```

➢ After that, we make the final formatting of the final file (lines of the file), enter the tabs and the numbering for the hierarchy and print it.

➢ Finally we delete every temporary file we have created.