**Faculty of Engineering and Technology**

**Department of Electrical and Computer Engineering**

**Linux Lab**

**ENCS313**

**Project 1 &Report**

**Shell Script Project**

**Prepared by:**

**Christina Saba 1201255**

**Pierre Backleh 1201296**

**Date:**

**May 15 2022**

**The following project is required to do some changes for a student’s record system saved in a text file**

* **The code is:**

echo "Enter the name of the file you want to read from" #asking the user to enter the name of file to read data from

read filename

[ ! -e $filename ]

check=$(echo $?) #making sure that the name entered is correct otherwise a message pops and says that the entered value is incorrect

if [ $check -eq 0 ]

then

echo "The file $filename does not exist"

exit 1

fi

cp $filename test.txt #copying the file to another temporary file

while true #showing the menu of choices for the user to choose from

do

echo " MAIN MENU "

echo " =========================================================="

echo "[1] Show student records for all semesters "

echo "[2] Show student records for a specific semester"

echo "[3] Show overall average"

echo "[4] Show average for every semester"

echo "[5] Show the total number of passed hours"

echo "[6] Show the precentage of total passed hours in relation to total F and FA hours"

echo "[7] Show total number of hours taken in every semester"

echo "[8] Show the total number of courses taken"

echo "[9] Show the total number of labs taken"

echo "[10] Insert the new semester record"

echo "[11] Change in course grade"

echo "[12] Exit program"

echo "==========================================================="

echo "Enter the number of operation you want to perfom"

read value

case "$value"

in

1) #the first choice is to display the records taken through the years in the text file

grep "EN" file.txt | tr -s ' ' ' ' | cut -d';' -f2 | tr ',' '\n' #grep command to get all the lines in the file of courses in each line after using the pipe and cutting the data

;;

##############################################################

2) #the second choice is to display the list of courses of the semester entered by the user

echo "Enter the semester to show the records"

read line

grep "$line" file.txt | tr -s ' ' ' ' | cut -d';' -f2 | tr ',' '\n' #grep the line which has the value entered by the user and cutting it to display the record

;;

###############################################################

3) #the third choice is to display the overall average of the whole semesters

avg=0

avgsum=0

hoursummation=0

mult=0

grep "EN" file.txt | tr -s ' ' ' ' | cut -d';' -f2 | tr ',' '\n' | grep -v "I" | sed 's/FA/50/' | sed 's/F/55/' | sort -r | uniq -w 9 | cut -d' ' -f3 > g.txt #separating the grades

grep "EN" file.txt | tr -s ' ' ' ' | cut -d';' -f2 | tr ',' '\n' | grep -v "I" | sed 's/FA/50/' | sed 's/F/55/' | sort -r | uniq -w 9 | cut -c7 > hour.txt #separating the hours using cut -c7

paste g.txt hour.txt > avg.txt # pasting the values to the same file to read from and calculate the average by multiplying each grade by its hour then computing the summation and divide them by the sum of the hours

files="avg.txt"

while read -r VAR1 VAR2; do

g=$VAR1

h=$VAR2

(( hoursummation += h ))

mult=$(( g \* h ))

(( avgsum += mult ))

done <$files

echo "The overall average is: $(bc -l <<<"${avgsum}/${hoursummation}")"

;;

################################################################

4) #the fouth choice is to display each semester's average separately

sed '1d' file.txt > test.txt

r="test.txt"

echo "Each semester has an average of:"

while IFS= read -r line; do

semhours=0

mul=0

average=0

echo $line | tr -s ' ' ' ' | cut -d';' -f2 | tr ',' '\n' | sed 's/FA/50/' | sed 's/F/55/' | cut -d' ' -f3 > semgrades.txt #putting the grades in a file

echo $line | tr -s ' ' ' ' | cut -d';' -f2 | tr ',' '\n' | cut -c7 > semhours.txt # putting the hours in a file

paste semgrades.txt semhours.txt > semavg.txt #pasting the values in the same file to compute the average

k="semavg.txt"

while read -r VAR1 VAR2; do

a=`echo $VAR1`

b=`echo $VAR2`

(( semhours += b ))

mul=$(( a \* b ))

(( average += mul ))

done <$k

echo $(bc -l <<<"${avgsum}/${hoursummation}")

done <$r

;;

#################################################################

5) #the fifth choice is to compute the total passed hours taken and without any repitition in courses

grep "EN" file.txt | tr -s ' ' ' ' | cut -d';' -f2 | tr ',' '\n' | grep -v "I" | grep -v "F" | grep -v "FA" | sort -r | uniq -w 9 | cut -c7 > hours.txt #taking the hours of every course that is unique and saving them in a file

file1="hours.txt"

psum=0

phour=0

while read -r line; do #while loop to compute the summation of the hours passed

phour=$line

(( psum += phour ))

done <$file1

echo "The total passed hours are: $psum"

;;

#################################################################

6) #the sixth choice is to find the precentage of F and FA divided by the summation of passed hours

grep "EN" file.txt | tr -s ' ' ' ' | cut -d';' -f2 | tr ',' '\n' | grep -v "I" | sort -r | uniq -w 9 | cut -d' ' -f3 > totalgrades.txt

grep "EN" file.txt | tr -s ' ' ' ' | cut -d';' -f2 | tr ',' '\n' | grep -v "I" | grep -v "F" | grep -v "FA" | sort -r | uniq -w 9 | cut -c7 > FA.txt

numFA=$(grep 'FA\|F' totalgrades.txt | wc -l) #couting the number of lines in the file FA.txt which has the hours as each line is represented by an hour

file2="FA.txt"

passsum=0

h=0

while read -r line; do

h=$line

(( psum += h ))

done <$file2

percentage=$(bc -l <<<"${numFA}/${psum}") #dividing the number of F and FA found by the total number of hours passed

echo "the percentage of F and FA is $percentage"

;;

#################################################################

7) #the seventh choice is to find the total taken hours of each semester seperately

sed '1d' file.txt > test.txt

f="test.txt"

echo "Each semester has a total taken hours of:"

while read -r line; do

sum=0

semhour=0

echo $line | tr -s ' ' ' ' | cut -d';' -f2 | tr ',' '\n' | cut -c7 > semhours.txt #cutting each line which represents a semester and saving the hours in a separate file

file3="semhours.txt"

while read -r line; do

semhour=$line

(( sum += semhour ))

done <$file3

echo $sum

done <$f

;;

#################################################################

8) #the eighth choice is to print the total number of hours taken of the unique courses

echo "Total number of courses taken is: $(grep "EN" file.txt | tr -s ' ' ' ' | cut -d';' -f2 | tr ',' '\n' | grep -v "I" | sort -r | uniq -w 9 | wc -l)"

;;

#################################################################

9) #the ninth choice is to find the total hours of lab courses in which the hour should be equal to 1

grep "EN" file.txt | tr -s ' ' ' ' | cut -d';' -f2 | tr ',' '\n' | grep -v "I" | sort -r | uniq -w 9 | cut -c7 > lab.txt

file4="lab.txt"

labcount=0

while read -r line; do

labhour=$line

if [ $labhour -eq 1 ] #making sure if the hour is 1 then the count is incremented by one

then

(( labcount += 1 ))

fi

done <$file4

echo "The number of labs taken is: $labcount"

;;

################################################################

10) #the tenth choice is to ask the user to enter a whole semester record with restriction on the inputs and if it was correct then add to the file of records

echo "Enter the record you want to add:"

read record

echo $record | tr -s ' ' ' ' | cut -d';' -f2 | tr ',' '\n' | cut -d' ' -f2 > recordcourses.txt

echo $record | tr -s ' ' ' ' | cut -d';' -f2 | tr ',' '\n' | grep -v "I" | sed 's/FA/50/' | sed 's/F/55/' | cut -d' ' -f3 > recordgrades.txt

echo $record | tr -s ' ' ' ' | cut -d';' -f2 | tr ',' '\n' | grep -v "I" | sed 's/FA/50/' | sed 's/F/55/' | cut -c7 > recordhours.txt

recordgrades="recordgrades.txt"

recordcourses="recordcourses.txt"

recordhours="recordhours.txt"

#reading the courses and making sure it begins with ENCS or ENEE

while read -r line; do

if [ `echo $line | grep -q "ENCS"` -o `echo $line | grep -q "ENEE"` ]

then

:

fi

if [ $? -ne 0 ]

then

echo "This record cannot be added"

break

fi

done <$recordgrades

#reading the grade and making sure each grade is between 60 and 99 or F or FA

while read -r line; do

if [ $line -gt 60 -a $line -lt 99 -o $line = "FA" -o $line = "F" -o $line = "I" ]

then

:

else

echo "This record cannot be added"

break

fi

done <$recordgrades

#reading the hours and making sure the summation is not less than 12

recordsummation=0

while read -r line; do

(( recordsummation += $line ))

done <$recordhours

if [ $recordsummation -lt 12 ]

then

echo "This record cannot be added"

break

fi

if [ $? -ne 0 ]

then

echo "This record cannot be added"

else

echo $semester >> file.txt

fi

;;

################################################################

11)#the eleventh choice is to change a chosen grade from the user and making sure the user wants to change if the course is found

echo "Enter the course code you are trying to change:"

read code

echo "Enter the new grade you are trying to provide:"

read newgrade

grep "EN" file.txt | tr -s ' ' ' ' | cut -d';' -f2 | tr ',' '\n' | sed 's/FA/50/' | sed 's/F/55/' > newcourses.txt

oldcourse=$(grep $code newcourses.txt)

echo "Old course is $oldcourse"

echo "Do you want to change the grade? (yes or no)"

read answer

if [ $answer = 'yes' ]

then

sed -i "s/$oldcourse/ $code $newgrade/g" file.txt #using the sed command to change the old value with a new oneS

echo "This grade is changed successfully"

else

echo "The grade is not changed"

fi

;;

################################################################

12) #the twelveth choice is to exit the program

echo "Exitting the program"

exit 0

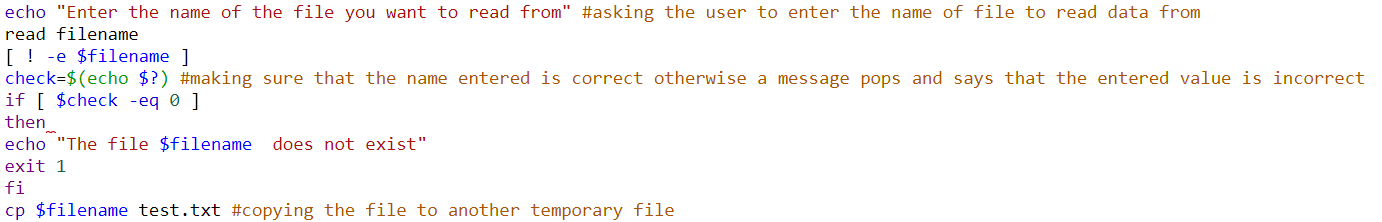
;;

################################################################

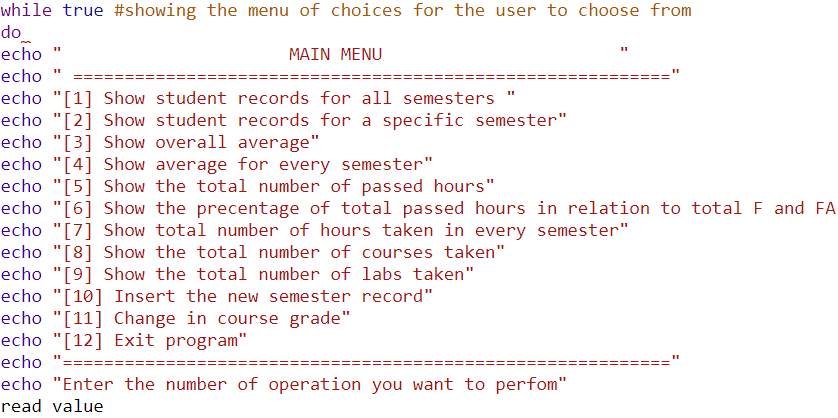
esac

done

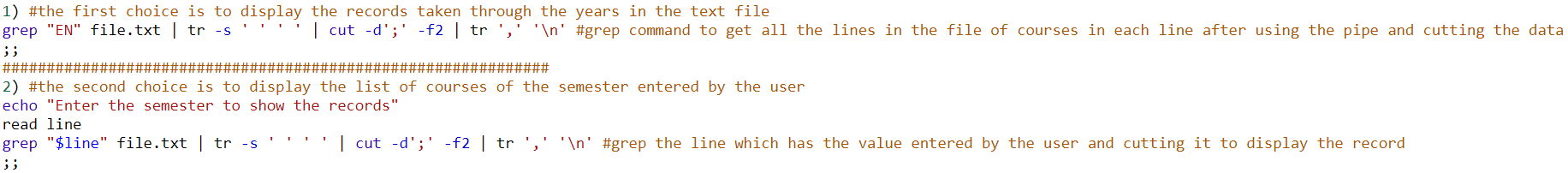
* **Screenshots from the previous code:**

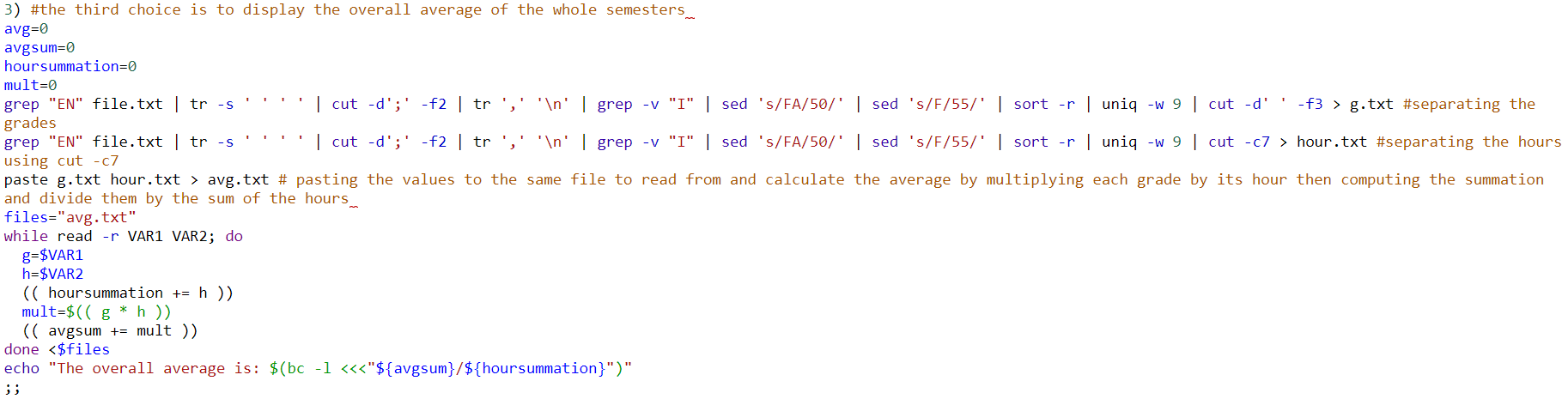


This previous code segment we selected the file to read data from and check if it exists or not. Then copying the file to a temporary one to use in the next cases. If the file is right then the code continues if not then a message displays that the file does not exist.

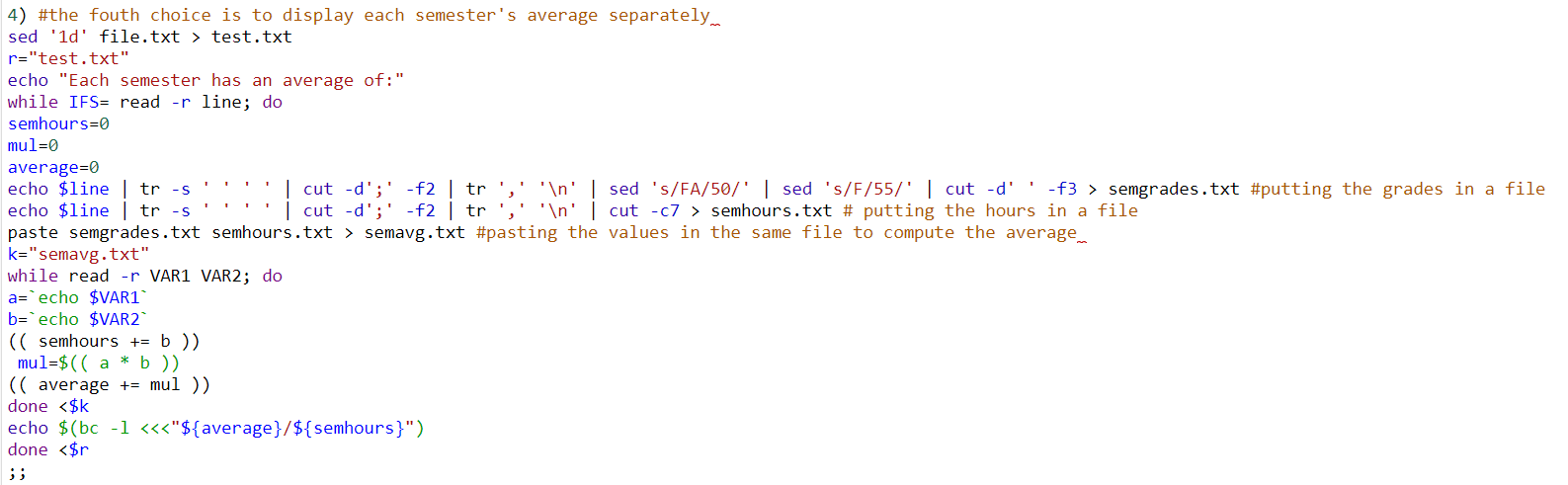


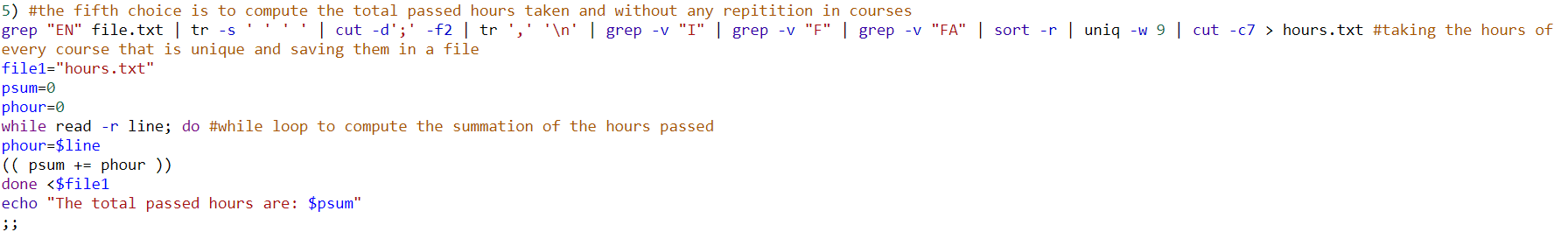
This segment shows the menu done to display the choices to the user to choose using case method and assigning each integer with a choice.



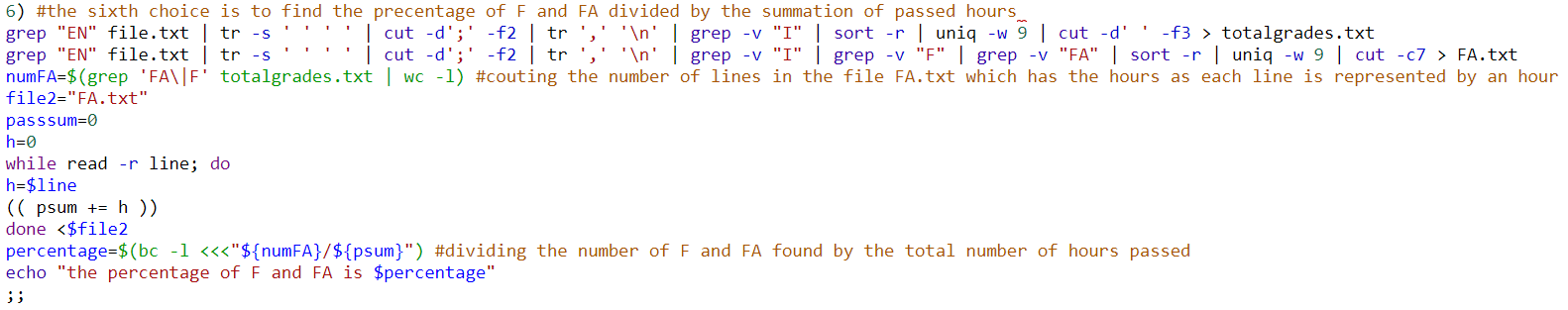
This segment code consists of two cases. The first one is to display all the courses through all semesters which is a pipe which searches for each line using grep command for “EN” in the file and using cut command cuts the line based on (;) then using translate command to change each comma separating the courses with a new line. the second case is to display the list of courses for a specific semester entered by the user then using the same method as the first case.

This code segment shows the third case in which computes the overall average. The first pipe command gets each line in the original file then cutting the line and separating the courses in each line then using sed comma and to change F grade to 55 and FA to 50 and removing I grade which are incomplete courses. Then sorting courses and making sure they are unique using sort and uniq commands then using cut command to cut based on space to get the grades of each course then saving them in a file called g.txt. the second pipe does the same as the above but the cut command differs as it takes the seventh character which is the hour of each course then saving them in a file called hour.txt. avg.txt is then used to paste the values in it then using while loop to read the two values from each line and calculating the average by (grade \* hour) / (sum of hours) then using echo to print the value and by using bc -l to make sure that the fractions occur.

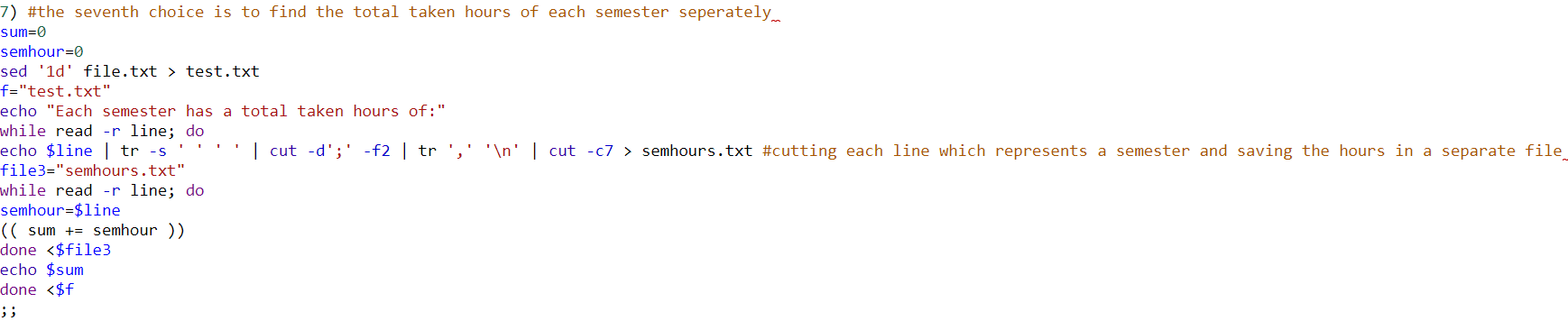


This code displays each semesters average separately. The same as the above code segment is done to separate the hours and grades then pasting them into another file to do the average computations using while loop but the difference is that another while loop is used to travel through each line of text.txt file which is the result of deleting the first line in the original file using sed ‘1d’ then using pipe command on each line which presents each semester and calculating the average and using bc -l to make sure that the fractions occur.

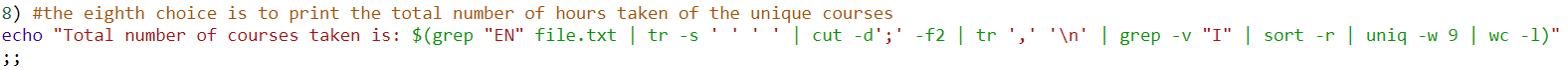
This segment is to compute the total passed hours taken without repetition in courses and the incomplete is removed along with F and FA courses using grep -v command to find the summation of the hours after saving them in hours.txt file and using while loop the sum is found the printed using echo.



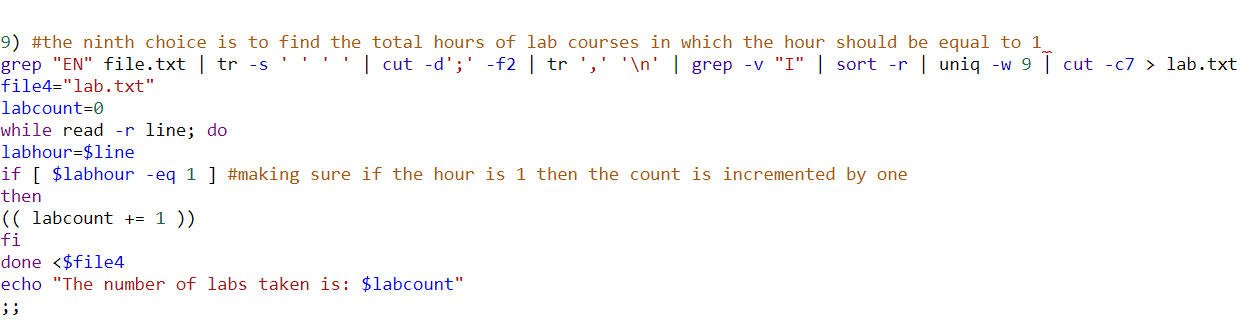
This segment is to divide the number of F and FA grades by the number of passed hours which are found the same way as the previous segment then using grep command then wc -l to count the number of lines which contain F or FA grades from a file of the unique grades totalgrades.txt then find the percentage using bc -l

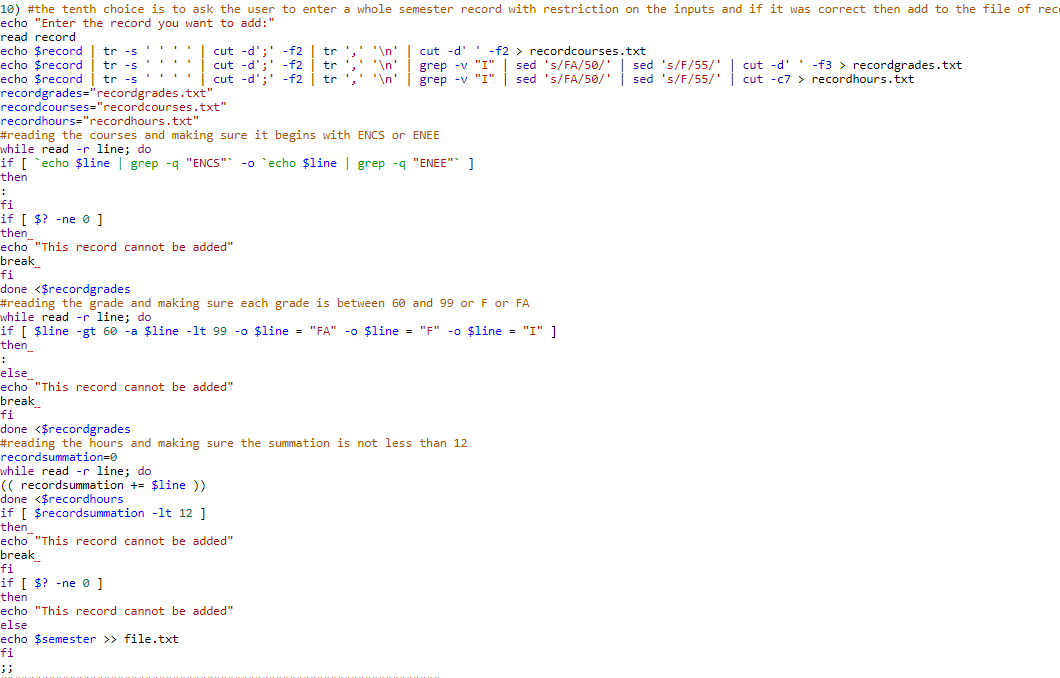


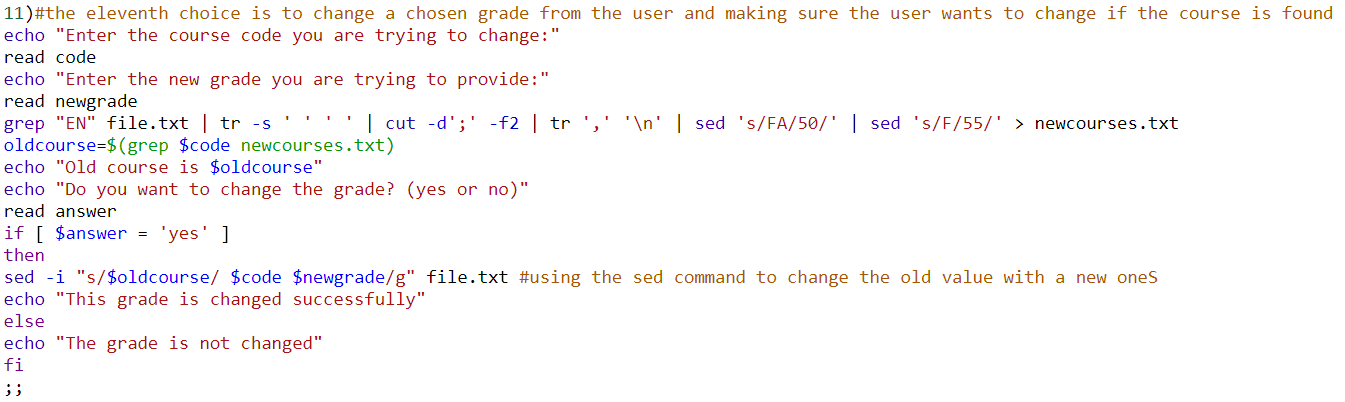
This segment prints the total number of hours in each semester. The same way as reading each semester’s grades and hours in one of the choices but the difference is that the cut command is used to cut the seventh character which is the hour of each course then using a while loop to find the summation of the hours and print them using echo.



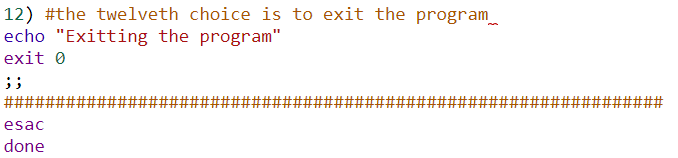
This segment code shows the total number of hours taken but only from the unique courses and excluding the incomplete courses by using a pipe command to cut each line and sort them to find the unique courses and then counting the lines using wc -l and print the value using echo.



This code segment finds the total number of lab courses through all semesters excluding the incomplete courses and making sure that there is no repetition in courses then the same way as getting the hours by using pipe and the cut command of the seventh character then using the while loop to make sure that if the hour is 1 then it is a lab so the count is incremented by 1 and the summation is then printed using echo.

This part is the tenth case which is adding a qualified record to the file by passing some restrictions and making sure if all the conditions are right then it can be added. This starts by using the pipe command to find the courses and save the in the file recordcourses.txt, then find the grades of each course and save into the file recordgrades.txt then saving the hours of each course in recordhours.txt. then making sure that the courses are either ENCS or ENEE using the grep command and if one is wrong then the record will not be added. Then checking the grades if they are between 60 and 99 or F or FA or I then they are qualified. Then finding the summation of the hours using the while loop and if it is less than 12 then the record is not available to be added.

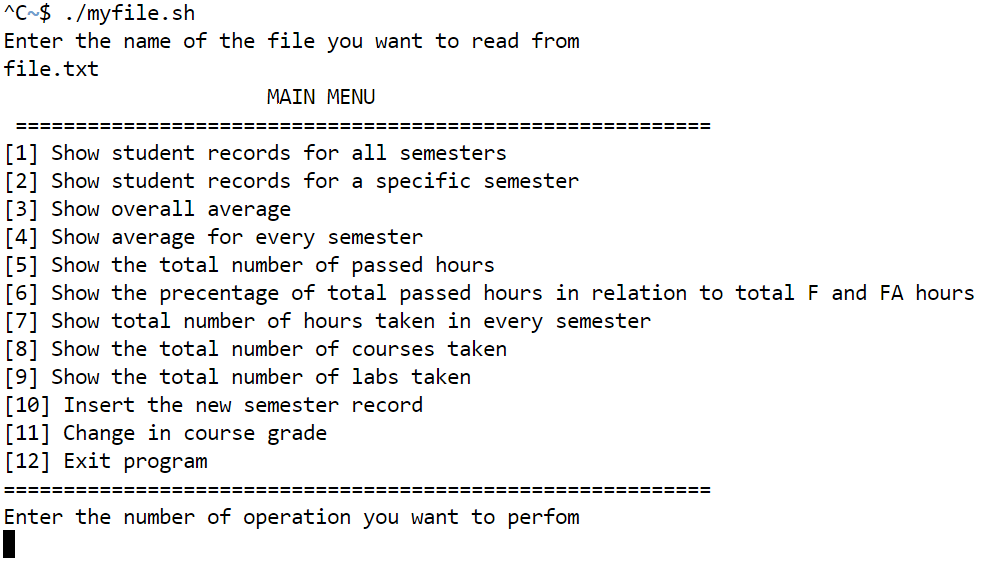
This segment code changes a grade in the orginal file by asking the user to enter the course code then the new grade to change then by using pipe command to separate the courses in another file to seach for the specific course then printing the old course and grade and then asking the user to ensure that the grade will be changed then if the answer is yes the sed command is used to place the old grade with the new one but if the answer is no then a message will pop that the grade is not changed.



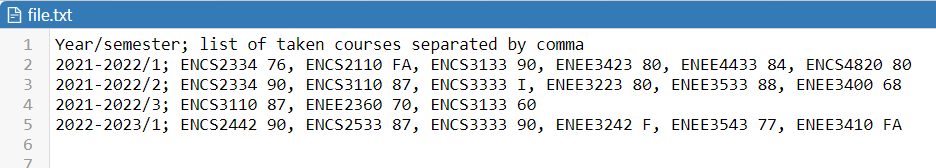
The last code segment is the last case to exit the program using exit 0 then ending the case with esac and ending the while loop with done.

**Testing the code on different cases:**

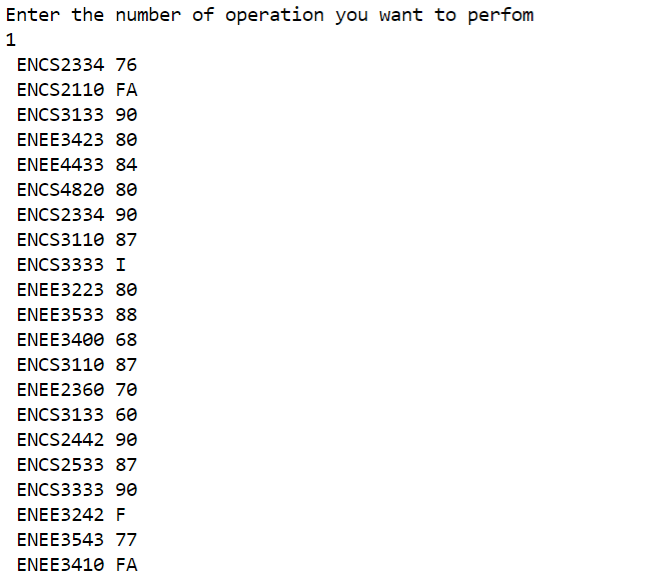
This is the main menu that appears when executing myfile.sh



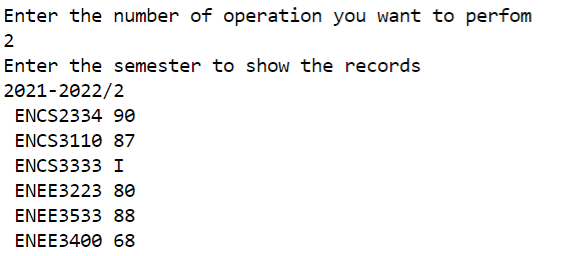
This is an example of the data in file.txt



Then by executing the first case: this shows the total courses and grades taken through all stored semesters in file.txt

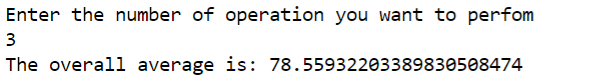


In the following picture the courses and grades are listed depending on the value enter by the user to determine which semester’s course can be given:

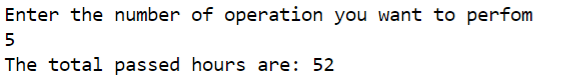


As it is seen the user entered the semester and the list of courses and grades are given below

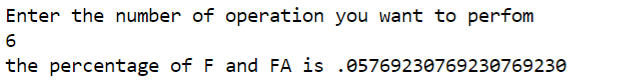
In the below picture the third case is executed and the overall average of all semesters is given



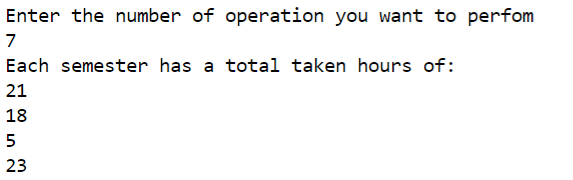
By executing the fifth case the total number of passed hours is given in the below picture:



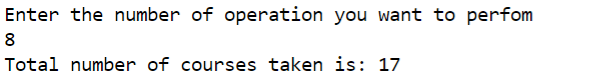
The following picture is the sixth case used to find the percentage of F and FA out of the total passed hours



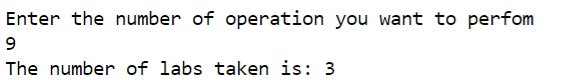
This following picture is the seventh case in which the total number of hours is found for each semester separately



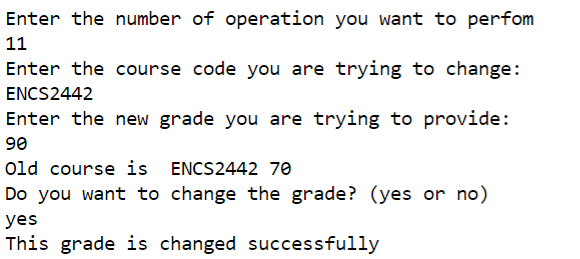
The following picture is the eighth case in which the number of non-repeated (unique) courses where the incomplete ones are executed:



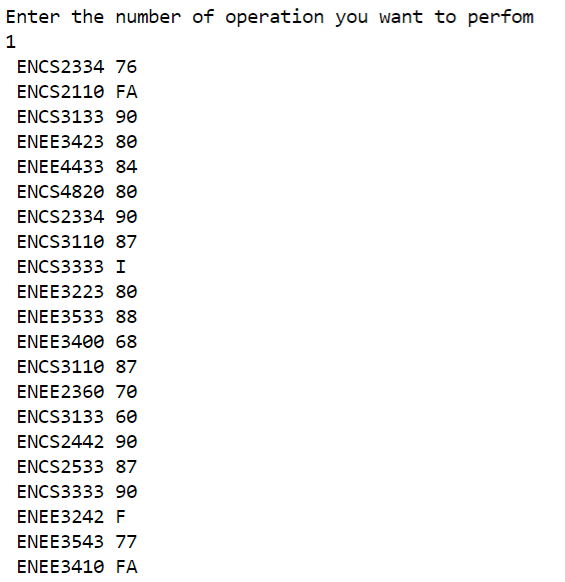
This picture is the ninth case in which the total number of labs taken depending on the hour which is supposed to be 1



The following picture is the tenth case which changes a given grade from the user to a specific course also entered by the user



As it’s seen the grade has been changed from 70 to 90 and to make sure the first case is executed and the change has been seen:



This is the last case to exit the program in which exit 0 was executed and successfully worked

