CSE333 - Operating Systems Project 1 Report

Install file 150114048_150115048_Pro1.zip and extract into a directory. In screenshots, you will see that the files extracted to Desktop/ directory. Then use commands consecutively;

:~$ cd Desktop/150114048_150115048_Pro1/menu // to open directory that myProgram.sh exists

:~$ chmod + myProgram.sh //change permissions to run myProgram.sh

:~$ ./myProgram.sh //run myProgram.sh

The commands written are also in first comment lines of myProgram.sh file.

After you will have the screen with 6 options, 5 code and 1 exit option, same as in Figure 1.

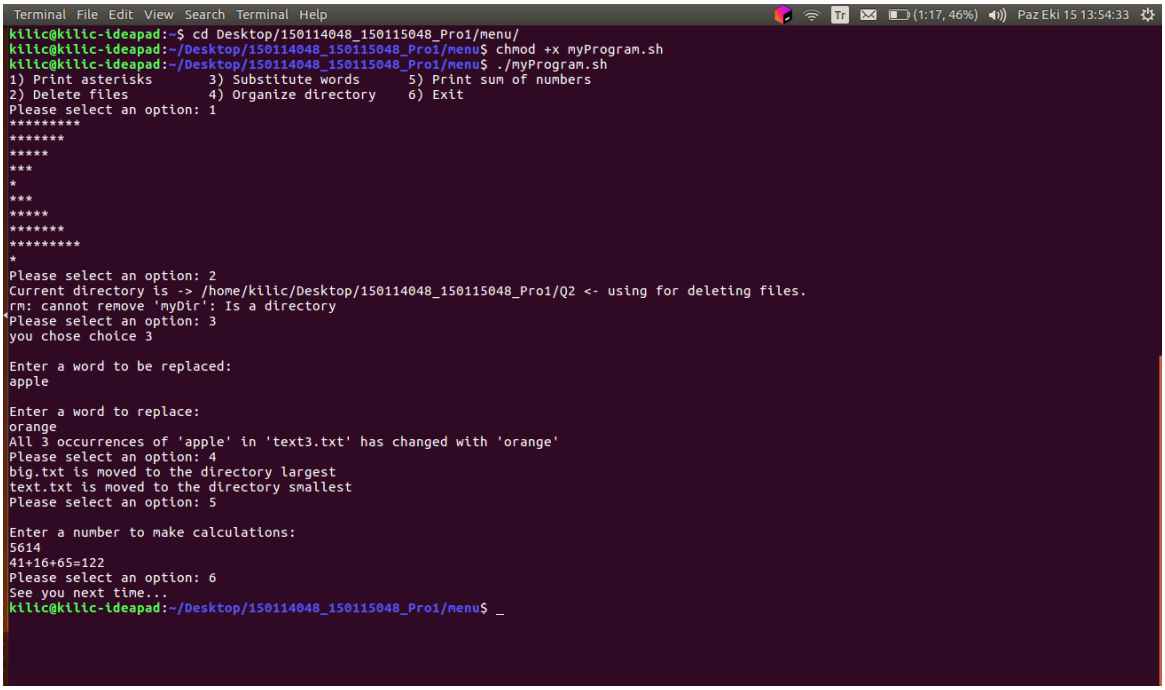

Figure 1

You can check all the options as shown in Figure 1.

If you want to see the changes while these shell script is running, you can check Figure 2,3,4,5,6 and 7 also you can check directory which have subdirectories menu,Q1,Q2,Q3,Q4 and Q5. In all of these directories, the given numbers related to question number. Check files with nano text editor or any other editor that support .sh file extension.

```
#!/bin/bash

# Author: Mehmet Cumali Demir <150114048>
# Author: Muhammed Kılıç <150115048>
#
# cd 150114048_150115048_Pro1/menu/
# Make this file executable:
# chmod +x myProgram.sh
#
# Run as
# ./myProgram.sh
# follow instructions

#print the menu interface
BASEDIR=$(pwd)
PS3='Please select an option: '
options=("Print asterisks" "Delete files" "Substitute words" "Organize directory" "Print sum of numbers"  "Exit")
select opt in "${options[@]}" #take option and evaluate
do
    case $opt in
        "Print asterisks")
                cd ..
                cd Q1/
                chmod +x *.*
                ./q1.sh "text1.txt"
                ;;
        "Delete files")
                cd ..
                cd Q2/
                chmod +x *.*
                ./q2.sh
                ;;

        "Substitute words")
                echo "you chose choice 3"
                cd ..
                cd Q3/
                chmod +x *.*
                printf "\nEnter a word to be replaced:\n"
                read var1
                printf "\nEnter a word to replace:\n"
                read var2
                ./q3.sh "text3.txt" $var1 $var2
                ;;
        "Organize directory")
                cd ..
                cd Q4/
                chmod +x *.*
                ./q4.sh
                ;;
        "Print sum of numbers")
                cd ..
                cd Q5/
                chmod +x *.*
                printf "\nEnter a number to make calculations:\n"
                read var
                ./q5.sh $var
                ;;
        "Exit")
                seq 1 15000 | while read i; do echo -en "\r$i"; done
                echo -en "\rSee you next time..."
                echo ""
            break
            ;;
        *) echo invalid option;;
    esac
done
```

Figure 2

18 lines (17 sloc) | 446 Bytes Raw Blame History

```
ARG_MISSING_ERROR=404
#file location for numbers
input="$1"
  if [ -z "$1" ] #check input
  then
     echo "argument is missing" #show error
  exit $ARG_MISSING_ERROR #exit with error code
  fi
        while IFS= read -r var
        do
                #counting for numbers to print$
                for (( number = 0 ; number< $var ; number++ ))
                do
                        printf "*" #symbol to print
                done
                printf "\n" #new line for next number
        done < "$input" #end of file read
```

Figure 3

20 lines (14 sloc) | 573 Bytes

Raw Blame History

```
IS_NOT_DIRECTORY=85 #error code
ARG1=$1 #first argument


if [[ -z "$1" ]]; then #there is no input use current directory

        echo "\nCurrent directory is -> $PWD <- using for deleting files.\n" #echo info

elif [[ -d $ARG1 ]]; then #if there is argument correctly use this
        echo "$ARG1 using"
    directory="$1"
        cd "$directory"

else
    echo "$ARG1 is not directory" #show error and exit with error code
    exit $IS_NOT_DIRECTORY
fi
        shopt -s extglob
        rm -f !(*.sh|*.c|"makefile"|"Makefile") #if there are files, remove all except that .c , .sh , makefile or Makefile.
```

Figure 4

36 lines (27 sloc) | 760 Bytes

Raw Blame History

```
ARG1_MISSING_ERROR=55 #error code
ARG2_MISSING_ERROR=56 #error code
ARG3_MISSING_ERROR=57 #error code
NOFILE=58 #error code
if [ -z "$1" ]
  then
    echo "3 arguments are missing" #show error
exit $ARG1_MISSING_ERROR
fi

if [ -z "$2" ]
  then
    echo "2 arguments are missing" #show error
exit $ARG2_MISSING_ERROR #exit with error code
fi

if [ -z "$3" ]
  then
    echo "1 arguments are missing" #show erro
exit $ARG1_MISSING_ERROR #exit with error code
fi

if [ ! -f "$1" ]  # Check  file exists.
then
  echo "File \"$1\" does not exist."
  exit $E_NOFILE
fi


sed -i -e "s/$2/$3/g" $1

 number_Of_Common=$(grep -o '\<'$3'\>' $1 | wc -l) #change word according to argument

echo "All $number_Of_Common occurrences of '$2' in '$1' has changed with '$3' "

```

Figure 5

22 lines (20 sloc) | 828 Bytes

Raw Blame History

```
#du -k * | sort -n
#du -k * | sort -n --reverse

#find the files just in the current directory and get the largest file
largestFile=`find . -maxdepth 1 -type f -exec basename {} \; | sort -n | head -1`
#find the files just in the current directory and get the smallest file
smallestFile=`find . -maxdepth 1 -type f -exec basename {} \; | sort -n --reverse | head -1`
#get the current directory
BASEDIR=$(pwd)
echo -en "\r"
#check if the directories exist
mkdir -p "$BASEDIR/"largest/
mkdir -p "$BASEDIR/"smallest/
#move the files to given directories
mv  "$BASEDIR/$largestFile" "$BASEDIR/"largest/
mv  "$BASEDIR/$smallestFile" "$BASEDIR/"smallest/
#echo the movements
echo "$largestFile is moved to the directory largest"
echo "$smallestFile is moved to the directory smallest"
#du -k * | sort -n
#du -k * | sort -n --reverse
```

Figure 6

42 lines (41 sloc) | 924 Bytes

Raw Blame History

```
#get argument
input="$1"
#remove zeros if exist at the beginning
number=$(echo $input | sed 's/^0*//')
#if the number is one digit case
if [ $number -lt 10 ]
then
        echo "$number=$number"
        exit 1
fi
#get the length of input
len=`echo ${#number}`
i=0
arrayOfDigits=""
#get the digits of input
while [ $number -gt 0 ]
do
        arrayOfDigits[$i]=$(( $number % 10 ))
        number=$(( $number / 10 ))
        i=$((i+1))
done
#variables to keep index
first=1
second=0
limit=$(($len-1))
sum=0
#print all of the new numbers and print numbers
while [ "$first" -lt "$len" ]
do
        if [ "$first" -lt "$limit" ]
        then
                echo -ne "${arrayOfDigits[$second]}""${arrayOfDigits[$first]}+"
        else
                echo -ne "${arrayOfDigits[$second]}""${arrayOfDigits[$first]}"
        fi
                let sum=$((sum+(($(${arrayOfDigits[$second]}*10))+$(${arrayOfDigits[$first]}))))))
                first=$(($first + 1))
                second=$(($second + 1))
done
#print sum of numbers
echo "=$sum"
```

Figure 7

The Questions numbered 2,3,4 are related to files and directories so in Figures 8 to 14 will show how these options works.

Question 2-)

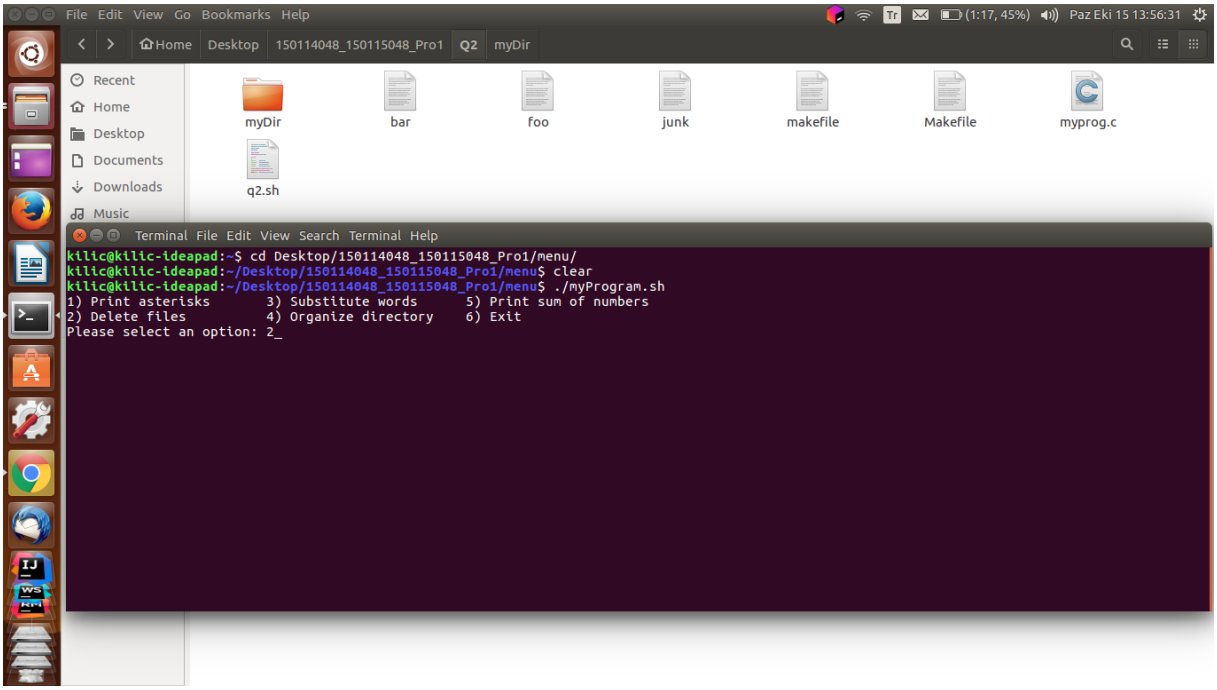

Figure 8

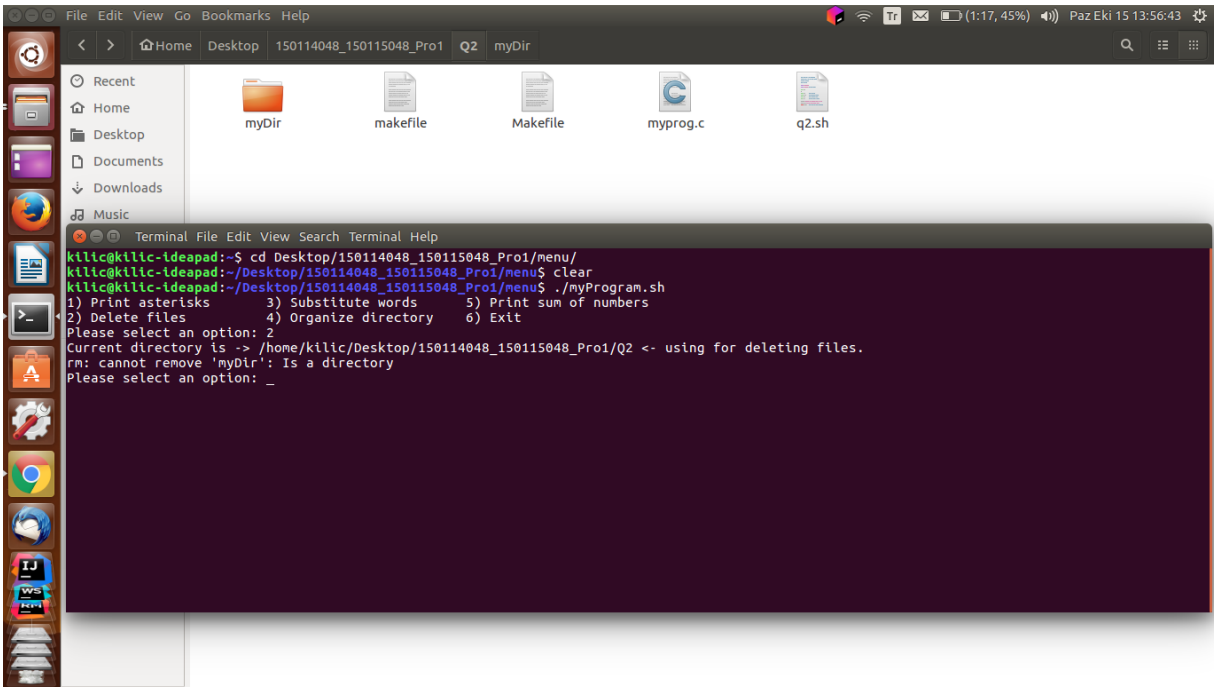

Figure 9

Question 3-)

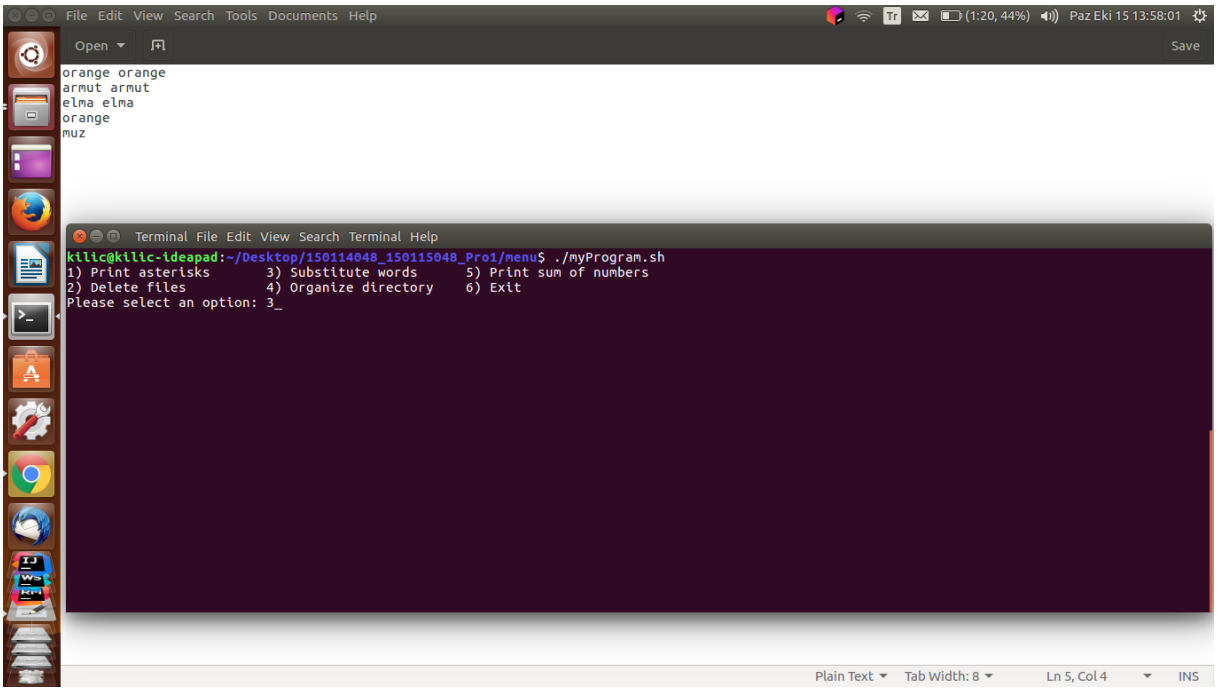

Figure 10

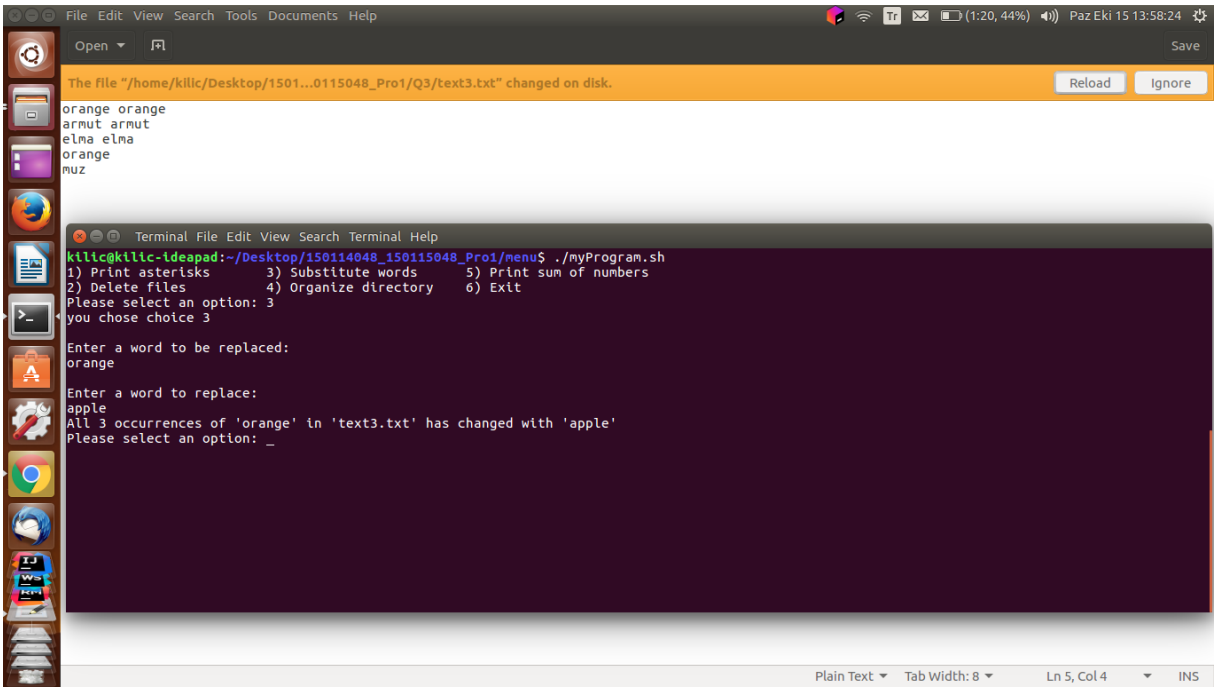

Figure 11

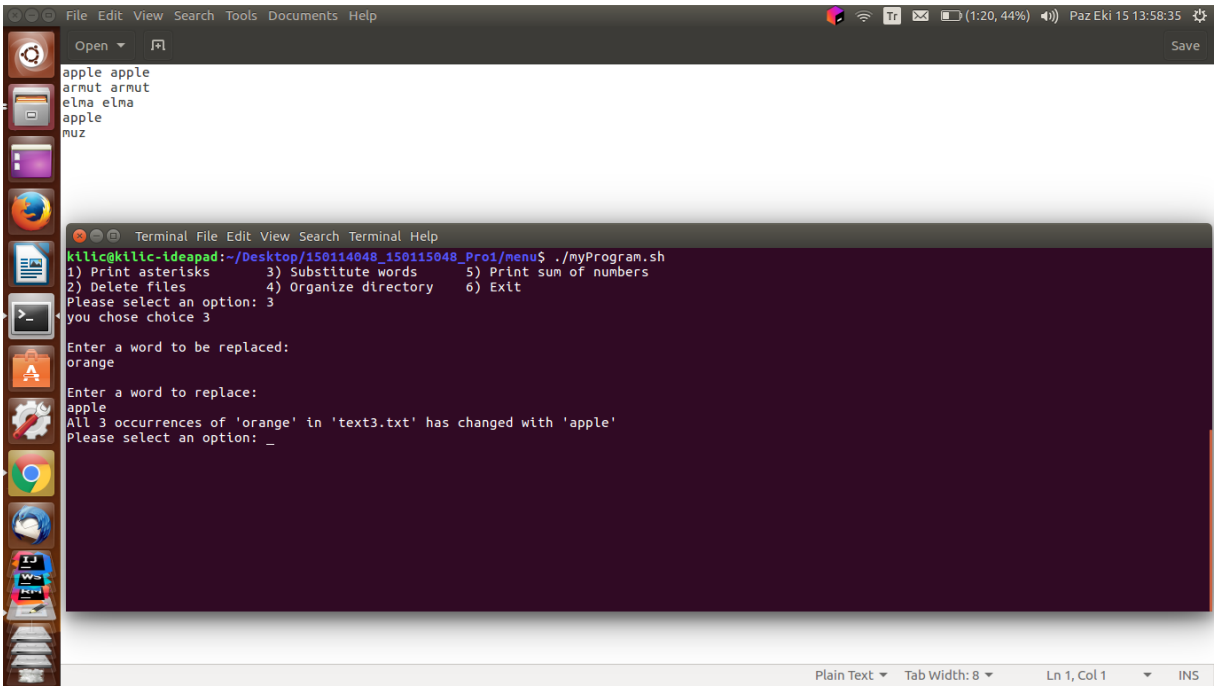

Figure 12

Question 4-)

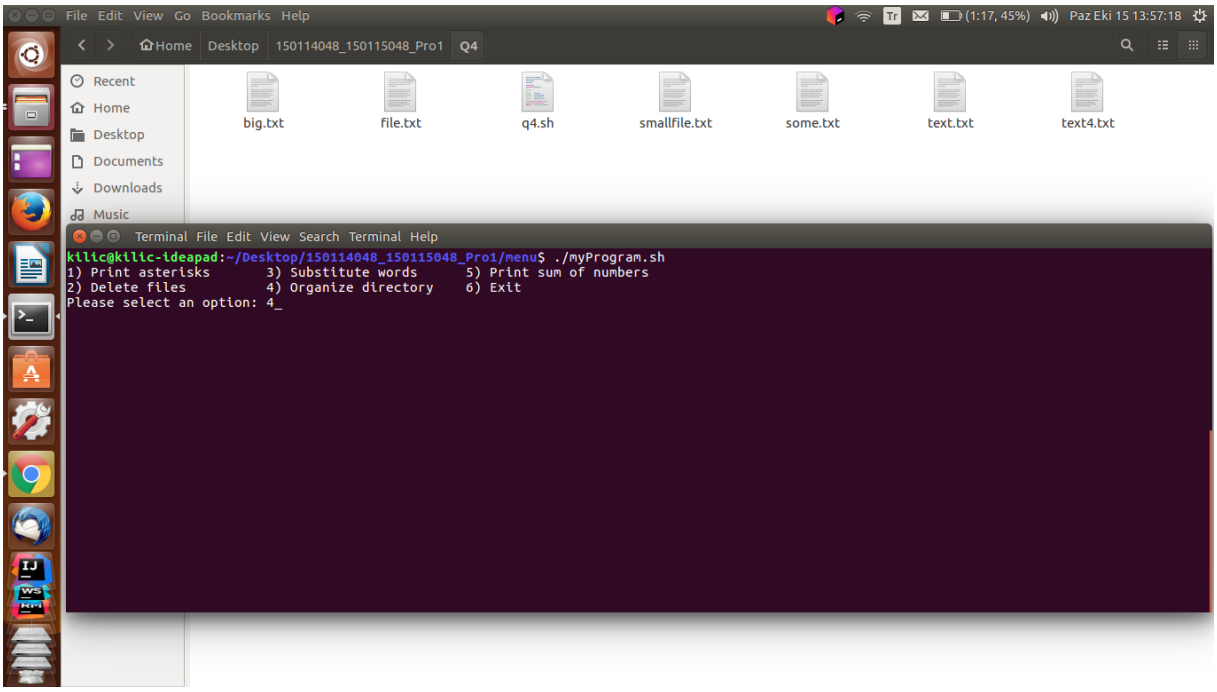

Figure 13

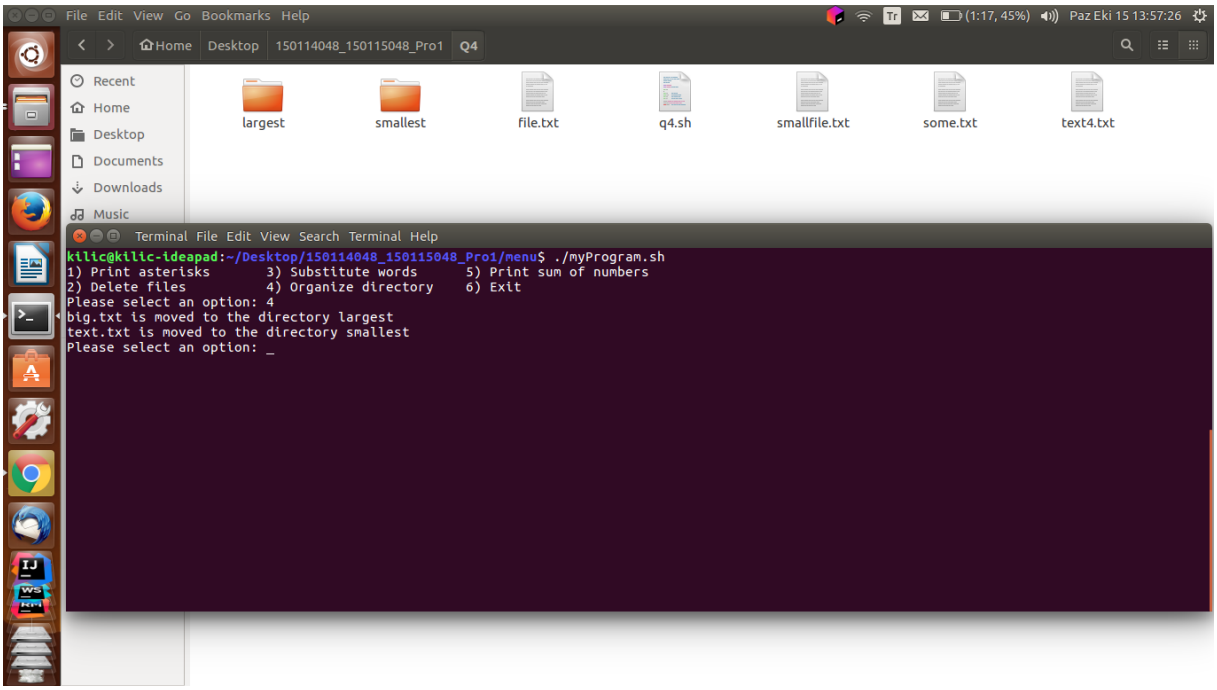

Figure 14

Technical Details

Q1-) Check that there is command line argument or not. If not exist prompt the error and exit with ARG_MISSING value. If argument exist, read text by line by and store the number value. Then execute for loop and print "*" value of each line.

Q2-) Check that there is command line argument or not in first if statement. If there is no argument, use current directory. Else if there is an argument check that is argument or not and then use rm command via –f (If file exist remove else do nothing.).

Q3-) Check that 3 command line argument has entered. If one is these are missing show suitable error and exit program with error code value. If every argument is available for command, then check if file exist check that there is file. The use grep function and send parameter input arg2 and arg3.

Q4-) Get the largest and smallest file names by commands and assign them to variables then check the directories to create directories if they are not exist. Move them to the given directories.

Q5-) Get a number as an argument, then separate this number to its digits. After assigning digits to and array, get the new numbers to calculate and show the sum.

Menu-) Switch case used in while to get options until user wants to exit the program. Firstly set chmod executable for all files in directory. Then execute relevant file. Then go previous directory via "cd.." . Then prompt menu if not pressed 6 and take an input from user. Then change directory into relevant input.