

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;

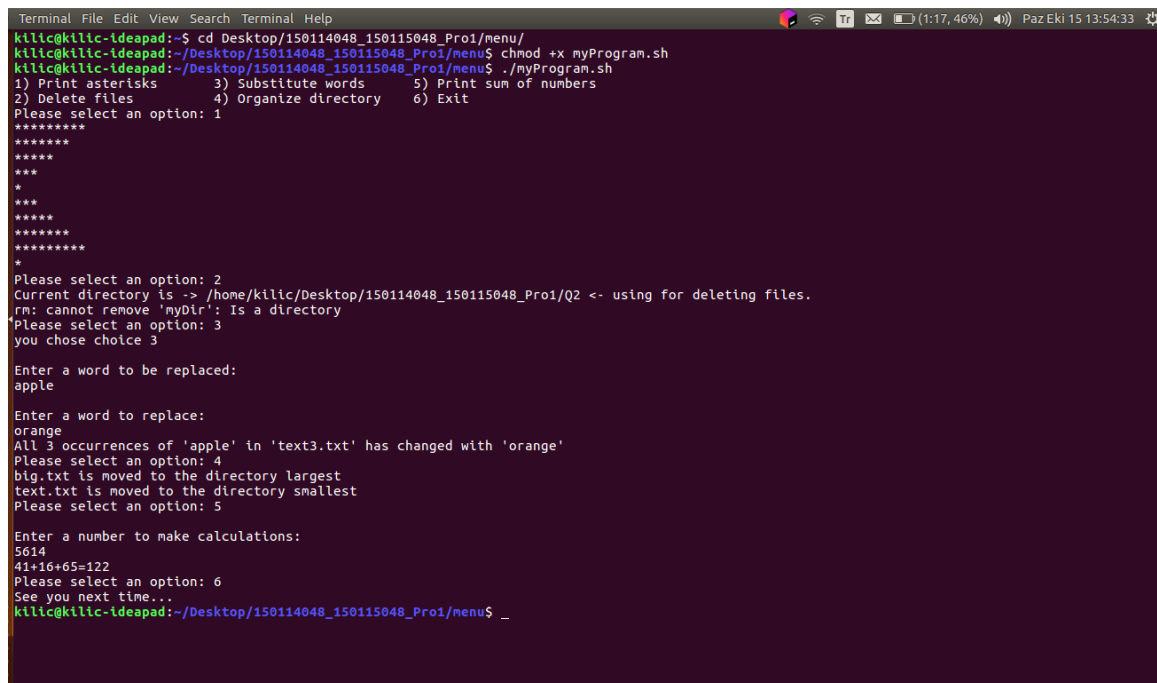
```
:~$ cd Desktop/150114048_150115048_Pro1/menu // to open directory that myProgram.sh exists
```

```
:~$ chmod +x 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.



```
Terminal File Edit View Search Terminal Help
kilic@kilic-ideapad:~$ cd Desktop/150114048_150115048_Pro1/menu/
kilic@kilic-ideapad:~/Desktop/150114048_150115048_Pro1/menu$ chmod +x myProgram.sh
kilic@kilic-ideapad:~/Desktop/150114048_150115048_Pro1/menu$ ./myProgram.sh
1) Print asterisks      3) Substitute words      5) Print sum of numbers
2) Delete files        4) Organize directory   6) Exit
Please select an option: 1
*****
*****
*****
***
*
***
*****
*****
*****
*
Please select an option: 2
Current directory is -> /home/kilic/Desktop/150114048_150115048_Pro1/Q2 <- using for deleting files.
rm: cannot remove 'myDir': Is a directory
Please select an option: 3
you chose choice 3
Enter a word to be replaced:
apple
Enter a word to replace:
orange
All 3 occurrences of 'apple' in 'text3.txt' has changed with 'orange'
Please select an option: 4
big.txt is moved to the directory largest
text.txt is moved to the directory smallest
Please select an option: 5
Enter a number to make calculations:
5614
41+16+65=122
Please select an option: 6
See you next time...
kilic@kilic-ideapad:~/Desktop/150114048_150115048_Pro1/menu$ _
```

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 Kiliç <150115048>
#
# cd 150114048 150115048 Proj/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

```

1 ARG_MISSING_ERROR=404
2 #file location for numbers
3 input="$1"
4 if [ -z "$1" ] #check input
5 then
6     echo "argument is missing" #show error
7     exit $ARG_MISSING_ERROR #exit with error code
8 fi
9 while IFS= read -r var
10 do
11     #counting for numbers to print$
12     for (( number = 0 ; number< $var ; number++ ))
13     do
14         printf "" #symbol to print
15     done
16     printf "\n" #new line for next number
17 done < "$input" #end of file read

```

Figure 3

```

20 lines (14 sloc) | 573 Bytes
1  IS_NOT_DIRECTORY=85 #error code
2  ARG1=$1 #first argument
3
4
5  if [[ -z "$1" ]]; then #there is no input use current directory
6
7      echo "\nCurrent directory is -> $PWD <- using for deleting files.\n" #echo info
8
9  elif [[ -d $ARG1 ]]; then #if there is argument correctly use this
10     echo "$ARG1 using"
11     directory="$1"
12     cd "$directory"
13
14  else
15     echo "$ARG1 is not directory" #show error and exit with error code
16     exit $IS_NOT_DIRECTORY
17  fi
18
19     shopt -s extglob
20     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
1  ARG1_MISSING_ERROR=55 #error code
2  ARG2_MISSING_ERROR=56 #error code
3  ARG3_MISSING_ERROR=57 #error code
4  NOFILE=58 #error code
5  if [ -z "$1" ]
6  then
7      echo "3 arguments are missing" #show error
8  exit $ARG1_MISSING_ERROR
9  fi
10
11  if [ -z "$2" ]
12  then
13      echo "2 arguments are missing" #show error
14  exit $ARG2_MISSING_ERROR #exit with error code
15  fi
16
17  if [ -z "$3" ]
18  then
19      echo "1 arguments are missing" #show erro
20  exit $ARG1_MISSING_ERROR #exit with error code
21  fi
22
23  if [ ! -f "$1" ] # Check file exists.
24  then
25      echo "File \"$1\" does not exist."
26      exit $E_NOFILE
27  fi
28
29
30  sed -i -e "s/$2/$3/g" $1
31
32  number_of_Common=$(grep -o '<'$3'>' $1 | wc -l) #change word according to argument
33
34  echo "All $number_of_Common occurrences of '$2' in '$1' has changed with '$3' "
35

```

Figure 5

```

22 lines (20 sloc) | 828 Bytes
1 #du -k * | sort -n
2 #du -k * | sort -n --reverse
3
4 #find the files just in the current directory and get the largest file
5 largestFile=`find . -maxdepth 1 -type f -exec basename {} \; | sort -n | head -1`
6 #find the files just in the current directory and get the smallest file
7 smallestFile=`find . -maxdepth 1 -type f -exec basename {} \; | sort -n --reverse | head -1`
8 #get the current directory
9 BASEDIR=$(pwd)
10 echo -en "\r"
11 #check if the directories exist
12 mkdir -p "$BASEDIR/"largest/
13 mkdir -p "$BASEDIR/"smallest/
14 #move the files to given directories
15 mv "$BASEDIR/$largestFile" "$BASEDIR/"largest/
16 mv "$BASEDIR/$smallestFile" "$BASEDIR/"smallest/
17 #echo the movements
18 echo "$largestFile is moved to the directory largest"
19 echo "$smallestFile is moved to the directory smallest"
20 #du -k * | sort -n
21 #du -k * | sort -n --reverse

```

Figure 6

```

42 lines (41 sloc) | 924 Bytes
1 #get argument
2 input="$1"
3 #remove zeros if exist at the beginning
4 number=$(echo $input | sed 's/^0*//')
5 #if the number is one digit case
6 if [ $number -lt 10 ]
7 then
8     echo "$number=$number"
9     exit 1
10 fi
11 #get the length of input
12 len=`echo ${#number}`
13 i=0
14 arrayOfDigits=""
15 #get the digits of input
16 while [ $number -gt 0 ]
17 do
18     arrayOfDigits[$i]=$(( $number % 10 ))
19     number=$(( $number / 10 ))
20     i=$((i+1))
21 done
22 #variables to keep index
23 first=1
24 second=0
25 limit=$((len-1))
26 sum=0
27 #print all of the new numbers and print numbers
28 while [ "$first" -lt "$len" ]
29 do
30     if [ "$first" -lt "$limit" ]
31     then
32         echo -ne "${arrayOfDigits[$second]}"${arrayOfDigits[$first]}+"
33     else
34         echo -ne "${arrayOfDigits[$second]}"${arrayOfDigits[$first]}"
35     fi
36     let sum=$((sum+(((${arrayOfDigits[$second]}*10))+(${arrayOfDigits[$first]})))
37     first=$((first + 1))
38     second=$((second + 1))
39 done
40 #print sum of numbers
41 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-)

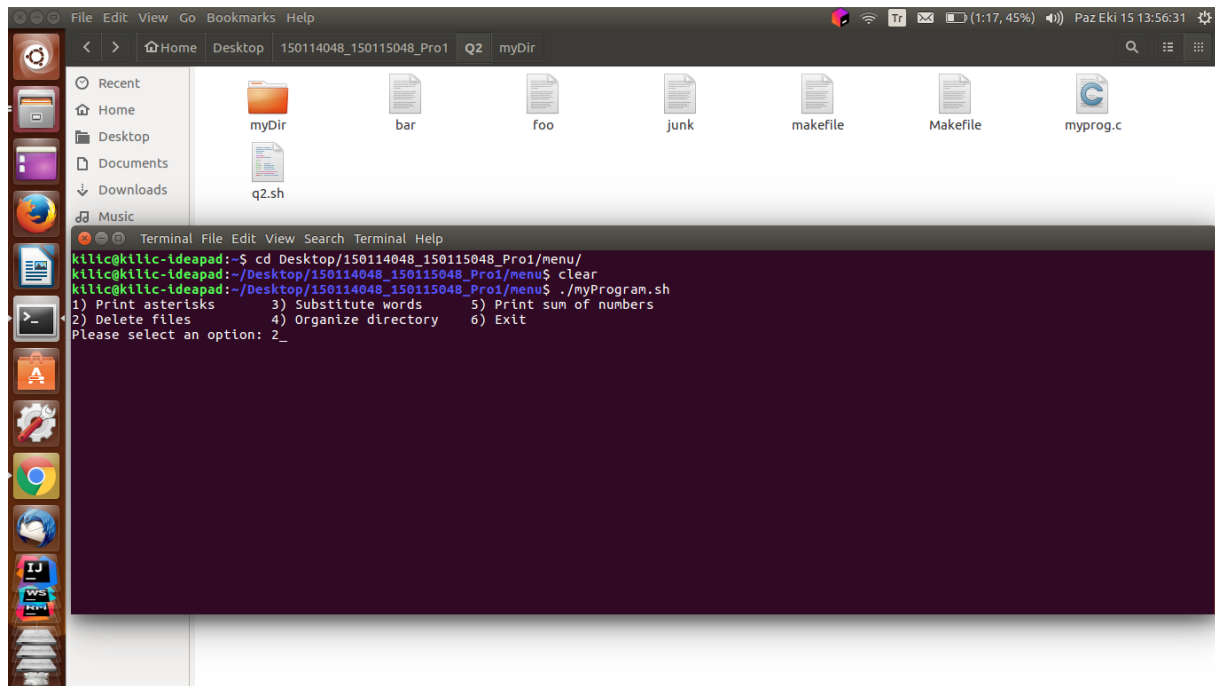


Figure 8

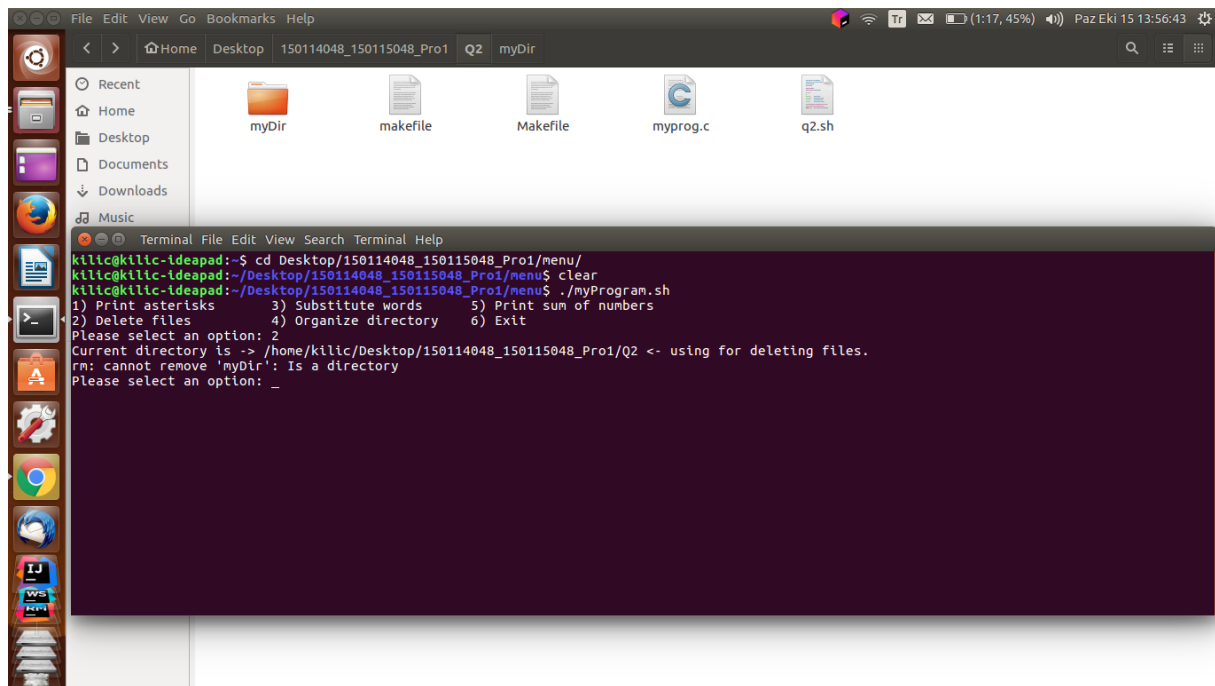


Figure 9

Question 3-)

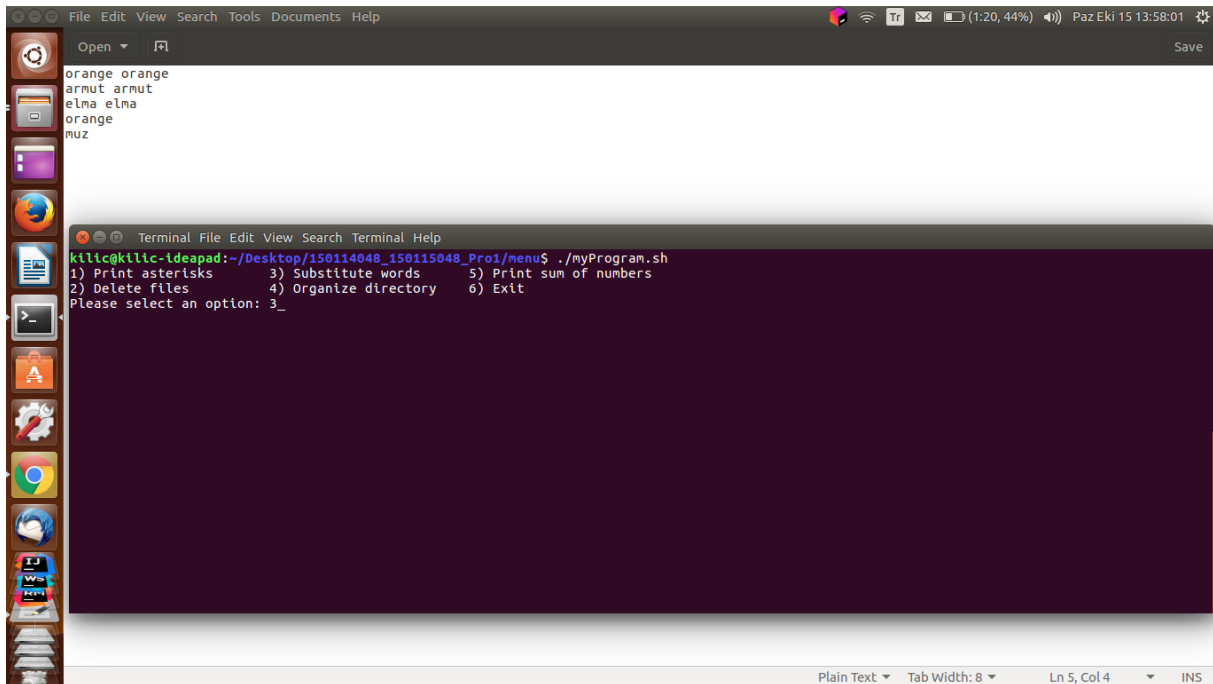


Figure 10

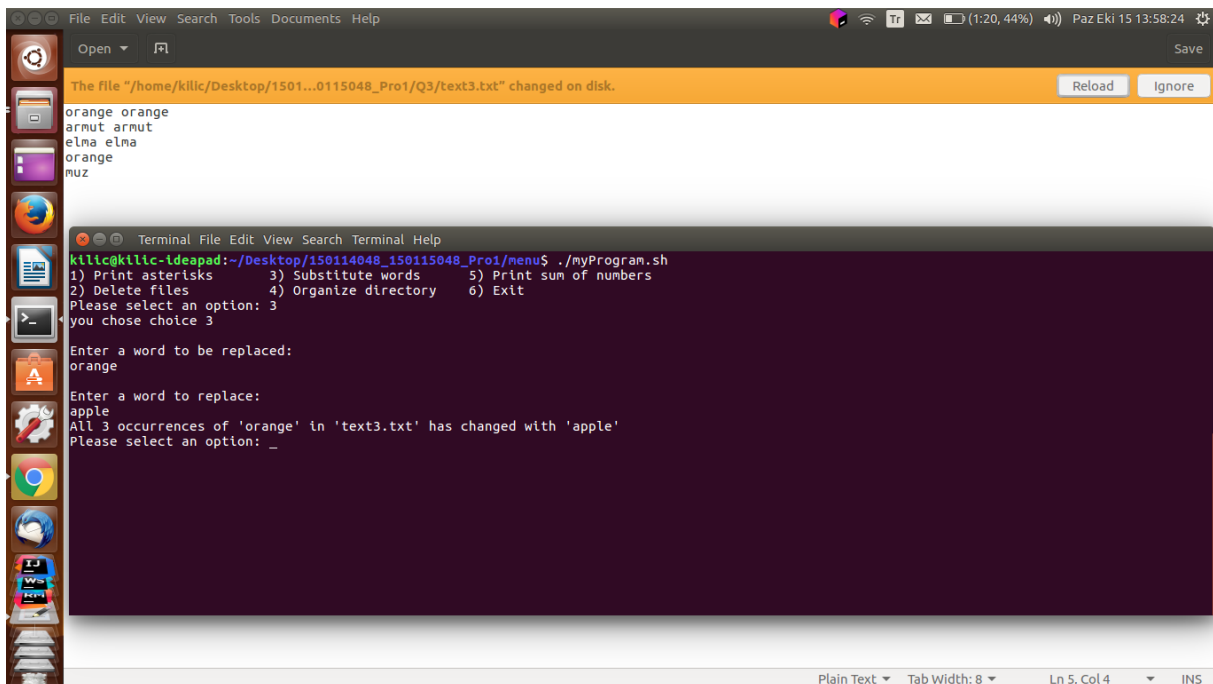


Figure 11

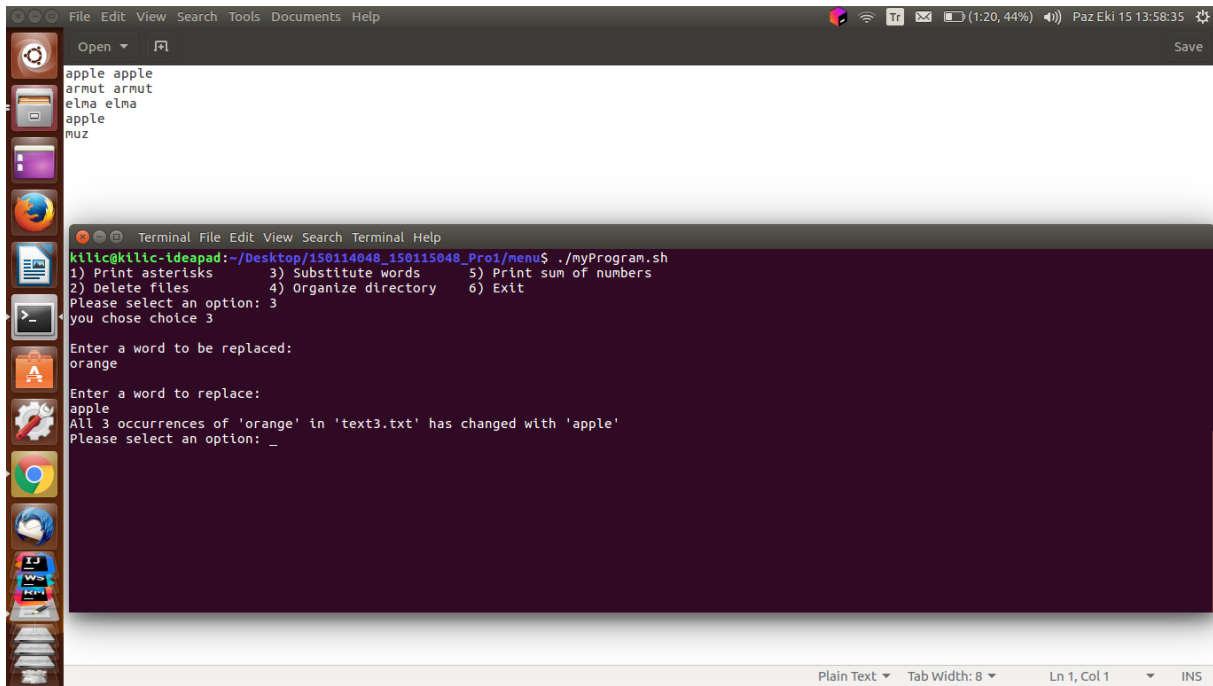


Figure 12

Question 4-)

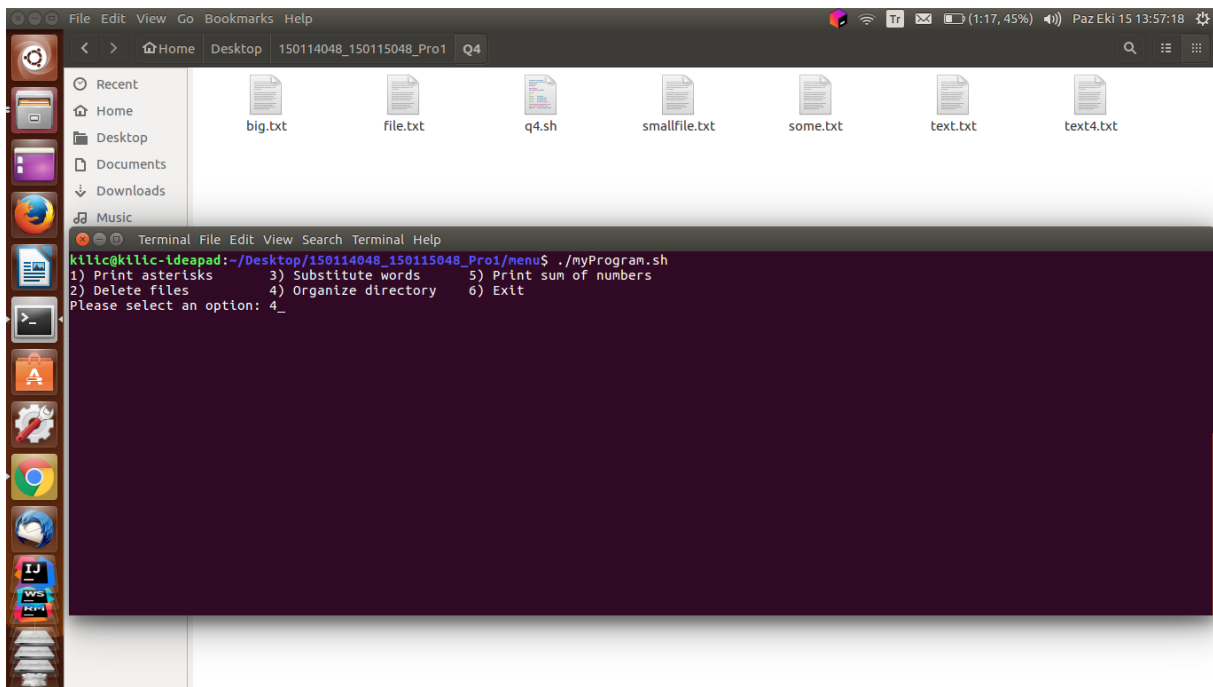


Figure 13

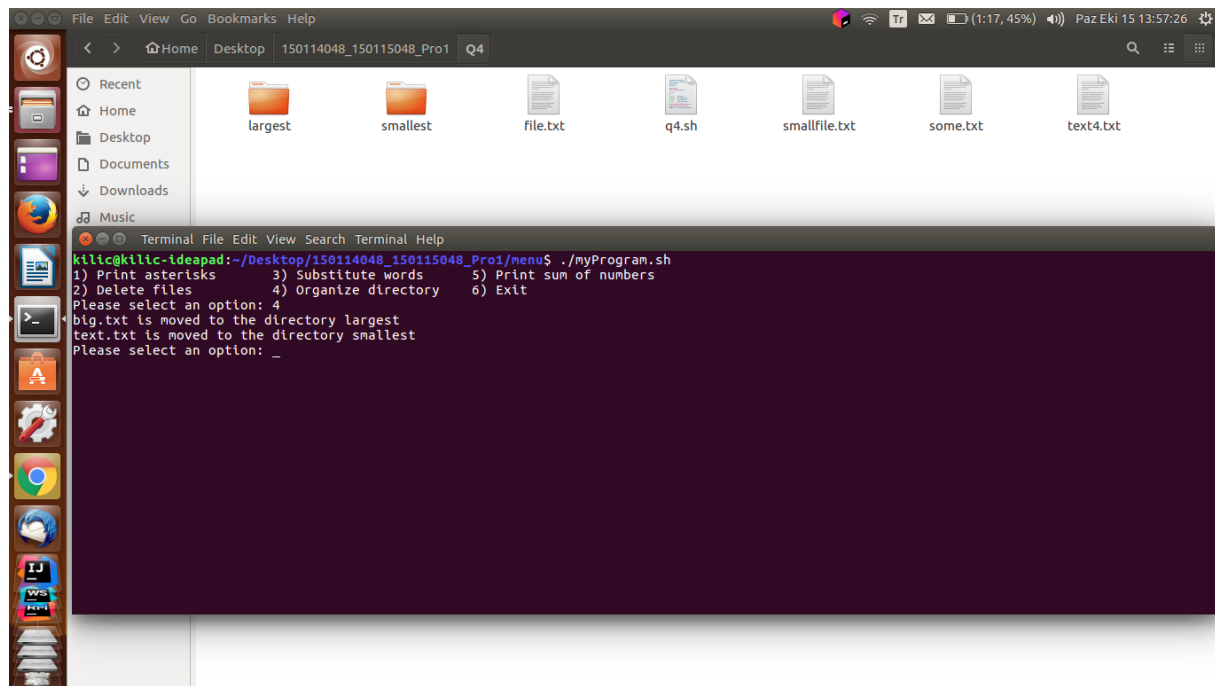


Figure 14

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 an 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.