BERZIET UNIVERSITY

Faculty of Engineering & Technology – Electrical & Computer Engineering Department

Summer Semester 2022

LINUX LABORATORY ENCS3130

Project1 Report (Shell Project)

Prepared by

Mahmoud Samara ID: 1191602

Mohammad AbuJaber ID: 1190298

Instructors

Dr. Aziz Qaroush

Dr. Mohammad Jubran

Date: 15th August 2022

Table of Contents

Table of Figures

1. Discussion The Project Code

1.1. The Main Menu

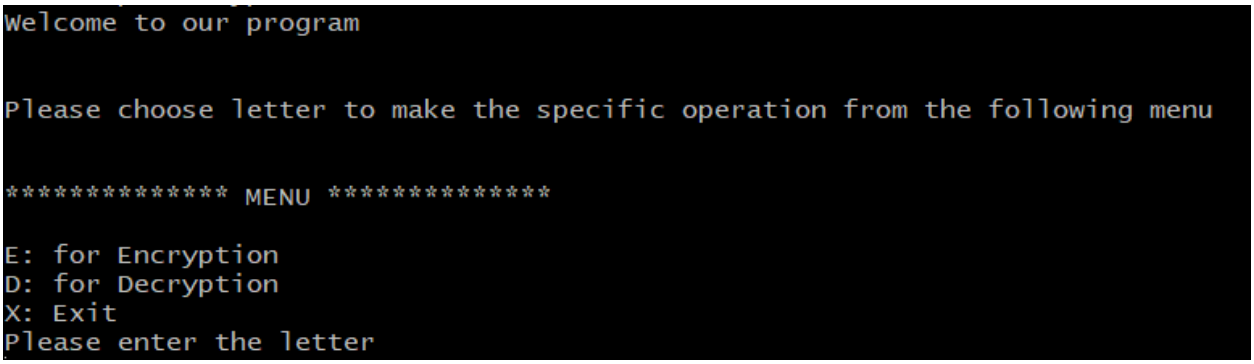

Figure 1: The Menu

This is our discussion about the shell project. In our project we will perform encryption and decryption for specific input files, whether they are words or binary numbers. In our code, first, the menu will appear for the user to choose whether he wants encryption operation or decryption. If the user entered a letter out of the menu, then the program asks the user to enter a valid value from the menu.

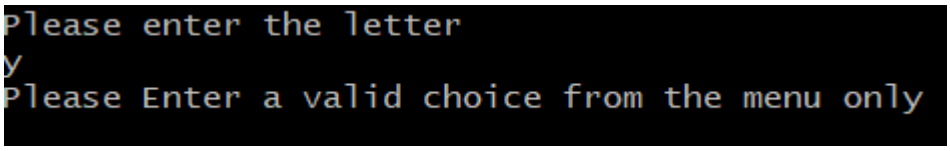

Figure 2: Entering a Non-valid Choice

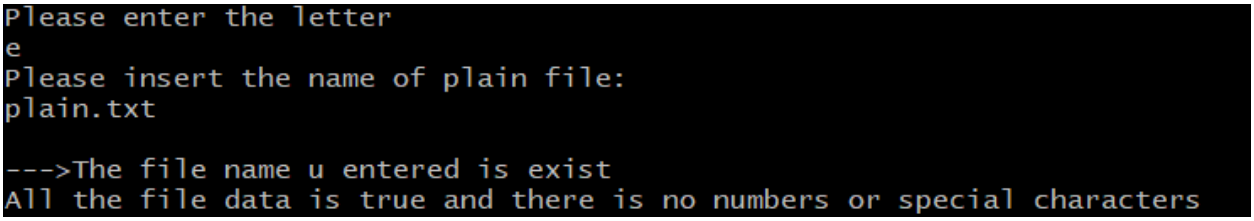

Figure 3: Entering an Existing File Name

For example, here the user chooses the encryption operation, so the program will ask the user to enter the name of the input file, then we will check if it exists or not. If the file exists, we will check if it contains only alphabets or not. If not, an error message will appear.

The code:

```
checkIfFileExist(){
while true # while statement that keep running until user enter an exist file
do
 if [ -e "$file" ] # if statement to check if the file is exist or not
 then
    printf "\n--->The file name u entered is exist\n"
    break
     else
printf "\n--->the file name that u entered is not exist , please enter right file name \n"
read file
   fi # end of if statement that check if file exist or not
done
}
```

Figure 4: Checking if the File Exist or Not

```
read file #reading the file name before entering if statement
checkIfFileExist
charcheck=$(cat $file | grep "[^A-Za-z ]") # check if the file contain any number or not

if [ -n "$charcheck" ]
   then
      printf " Error ==> There is at least a number or special character in your input file please check your file and get back to the program\n"
      exit
    else
      printf "All the file data is true and there is no numbers or special characters\n"
fi # end of if statement that check if there is any character
```

Figure 5: Checking if the File Contains Prohibited Characters

```
while true #while loop to keep reading the choice until the user enter E
do # beginning of while loop

read enteredletter  # reading the entered choice to know what case to do

case $enteredletter in

E|e)Encryption;;
D|d)Decryption;;
X|x)exit 0;;
*)echo "Please Enter a valid choice from the menu only";;

esac # end of case statement
done
```

Figure 6: While Loop to Keep the Program Running

Input example:

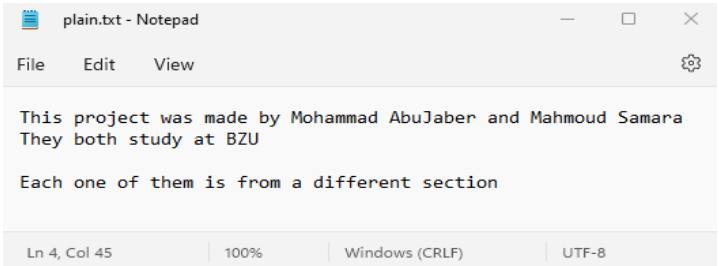

Figure 7: Input Example

1.2. Encryption

```
Please enter the letter
e
Please insert the name of plain file:
plain.txt

--->The file name u entered is exist
All the file data is true and there is no numbers or special characters
The Key In Decimal=  89
The Key In Binary=  01011001
The Encrypted Text=
11010000  00010011  00000011  10100010  01100000  10010010  10110010  01100011
00110011  11000011  10100011  11010010  01100000  11100010  10000011  10100010
01100000  01000011  10000011  11010011  11000011  01100000  10110011  00000010
01100000  01000001  01100011  00010011  10000011  01000011  01000011  10000011
11010011  01100000  10000001  10110011  11000010  00110001  10000011  10110011
11000011  10110010  01100000  10000011  01110011  11010011  01100000  01000001
10000011  00010011  01000011  01100011  11000010  11010011  01100000  10100000
10000011  01000011  10000011  10110010  10000011  10010001  11010000  00010011
11000011  00000010  01100000  10110011  01100011  11010010  00010011  01100000
10100010  11010010  11000010  11010011  00000010  01100000  10000011  11010010
01100000  10110001  00110000  11000000  10010001  10010001  11000001  10000011
10100011  00010011  01100000  01100011  01110011  11000011  01100000  01100011
11110011  01100000  11010010  00010011  11000011  01000011  01100000  00000011
10100010  01100000  11110011  10110010  01100011  01000011  01100000  10000011
01100000  11010011  00000011  11110011  11110011  11000011  10110010  11000011
01110011  11010010  01100000  10100010  11000011  10100011  11010010  00000011
01100011  01110011  10010101


The Encrypted text was saved in a file called cipher.txt
```

Figure 8: Encryption Output

After the program checked everything in the input file to be sure that it was as required, the program will now start calculating the key of the input file. The idea of our code is to use an array and put the word in it and loop for all the characters in this word. After that, we find the repeat of a character in the word and multiply it by the number of times it is repeated. Finally, we know that the word ends when space enters the for loop. The sum for each word will be calculated, and the sum will be mod 256. Now, after we found the key for each word, we put them in an array and compare all the numbers together to find the MAX value to print it as the key.

The code:

```
Encryption(){
cat /dev/null > cipher.txt
echo "Please insert the name of plain file:"
read file #reading the file name before entering if statement
checkIfFileExist
charcheck=$(cat $file | grep "[^A-Za-z ]") # check if the file contain any number or not

if [ -n "$charcheck" ]
   then
      printf " Error ==> There is at least a number or special character in your input file please check your file and get back to the program\n"
      exit
    else
      printf "All the file data is true and there is no numbers or special characters\n"
fi # end of if statement that check if there is any character
declare -a arrayOfMods
counter=0

letterIndex=0
ALPHABET=(Aa Bb Cc Dd Ee Ff Gg Hh Ii Jj Kk Ll Mm Nn Oo Pp Qq Rr Ss Tt Uu Vv Ww Xx Yy Zz)

for i in $(cat $file | tr " " "\n")
do
    for j in ${ALPHABET[@]}
    do
        ((++letterIndex))
        if [ "$(echo $i | grep [$j])" ]
        then
            repeat=$(echo $i | tr -cd [$j] | wc -c)
            ((sum+=letterIndex*repeat))
            arrayOfMods[$counter]=$(($sum%256))
        fi
    done
counter=$(($counter+1))
letterIndex=0
sum=0
done
key=${arrayOfMods[0]}
for k in ${arrayOfMods[@]}
do
    if [ $k -gt $key ]
    then
        key=$k
    fi
done
echo "The Key In Decimal= " $key
x={0..1}{0..1}{0..1}{0..1}{0..1}{0..1}{0..1}{0..1}
binaryKey=${x[$key]}
echo "The Key In Binary= " $binaryKey
```

Figure 9: Reading the File and Calculating the Key in Binary and Decimal

After that, to make the encrypted text message ready, we will make the XOR operation between the key value in decimal and each character ascii value in decimal. The XOR result will be converted to binary. After we made the XOR for all chars, we made the last step, which is to swap the first four bits with the last four bits for each character. In this step, we used the cut command, so we can choose as we like. Now the encrypted text message is ready and saved in the cipher file to be used as an input file for the decryption operation. All new lines '\n' were replaced by '@' and all spaces were replaced by '_' using (tr) command so that the program will encrypt each one of them into a 4-bit value and will be returned to same when decrypting the text file. Finally, the used arrays were set to null to be ready for another input file without problems.

```
##########  XOR with Swap operations ###############

declare -a arrayOfChars
declare -a arrayOfXORs
counter=0
convertToBinary(){
x=({0..1}{0..1}{0..1}{0..1}{0..1}{0..1}{0..1}{0..1})
binaryKey=${x[$tmp]}
arrayOfXORs[$counter]=$binaryKey
counter=$(($counter+1))
}
for i in $(cat $file | tr "\n" "@" | tr " " "_")
    do
    echo $i | sed 's/\(.\)/\1\n/g' >> temp_$$
done
counter=0
for j in $(cat temp_$$)
    do
    arrayOfChars[$counter]=$j
    ((counter++))
done

rm temp_$$

for k in ${arrayOfChars[@]}
    do
    case $k in
    A)tmp=$((65^$key))
    convertToBinary
    ;;
```

```
echo "The Encrypted Text= "
for l in ${arrayOfXORs[@]}
    do
    firstPart=$(echo -n $l | cut -c1-4)
    secondPart=$(echo -n $l | cut -c5-8)
    echo -n $secondPart$firstPart " "
    echo -n $secondPart$firstPart " " >> cipher.txt
done

x=({0..1}{0..1}{0..1}{0..1}{0..1}{0..1}{0..1}{0..1})
binaryKey=${x[$key]}
firstPart=$(echo -n $binaryKey | cut -c1-4)
secondPart=$(echo -n $binaryKey | cut -c5-8)
echo $secondPart$firstPart
echo -n $secondPart$firstPart " " >> cipher.txt

printf "\n\nThe Encrypted text was saved in a file called cipher.txt"
arrayOfXORs=()
arrayOfMods=()
arrayOfChars=()
menu
```

Figure 10: XOR with Swap Operations

1.3. Decryption

```
Please enter the letter
d
Please insert the name of cipher.txt file:
cipher.txt

--->The file name u entered is exist
The key in binary is:  01011001
The key in decimal is:  89


The swaped cipher.txt is:
00001101 00110001 00110000 00101010 00000110 00101001 00101011 00110110 00110011 00111100 0011101
0 00101101 00000110 00101110 00111000 00101010 00000110 00110100 00111000 00111101 00111100 00000
110 00111011 00100000 00000110 00010100 00110110 00110001 00111000 00110100 00110100 00111000 001
11101 00000110 00011000 00111011 00101100 00010011 00111000 00111011 00111100 00101011 00000110 0
0111000 00110111 00111101 00000110 00010100 00111000 00110001 00110100 00110110 00101100 00111101
 00000110 00001010 00111000 00110100 00111000 00101011 00111000 00011001 00001101 00110001 001111
00 00100000 00000110 00111011 00110110 00101101 00110001 00000110 00101010 00101101 00101100 0011
1101 00100000 00000110 00111000 00101101 00000110 00011011 00000011 00001100 00011001 00011001 00
011100 00111000 00111010 00110001 00000110 00110110 00110111 00111100 00000110 00110110 00111111
00000110 00101101 00110001 00111100 00110100 00000110 00110000 00101010 00000110 00111111 0010101
1 00110110 00110100 00000110 00111000 00000110 00111101 00110000 00111111 00111111 00111100 00101
011 00111100 00110111 00101101 00000110 00101010 00111100 00111010 00101101 00110000 00110110 001
10111 01011001


The Decrypted text was saved in a file called decryptionOutput.txt
```

Figure 11: Decryption Output

If the user chooses the decryption operation, first the key value will be printed. In the decryption, the key will be the last 8 bits of the cipher file after swapping them. We will use the for loop and the array idea with cut to print to make a swap for each 8 numbers in the file as the encryption idea. After we finished the swap operation, we made the XOR between the key and each of the 8 numbers. Finally, to make the decryption file ready for each XOR result, we found the decimal value of it and, using a case statement instead of each number, we put an alphabet char with '@' and '_' that are special cases in our code as we described previously in the encryption. The result will be printed in the output file.

The code:

```
Decryption(){
cat /dev/null > decryptionOutput.txt
echo "Please insert the name of cipher.txt file:"
read file #reading the file name before entering if statement
checkIfFileExist
Lasteight=$(awk '{print $NF}' $file) # get the last eight bits of the file
firstPart=$(echo -n $Lasteight | cut -c1-4)
secondPart=$(echo -n $Lasteight | cut -c5-8)
allEight=$secondPart$firstPart
echo "The key in binary is: " $allEight
echo "The key in decimal is: " $((2#$allEight))

declare -a arrayOfSwaped
counter=0
for i in $(cat $file)
do
    firstPart=$(echo -n $i | cut -c1-4)
    secondPart=$(echo -n $i | cut -c5-8)
    allEight=$secondPart$firstPart
    arrayOfSwaped[$counter]=$allEight
    ((counter++))
done
printf "\n\nThe swaped cipher.txt is:\n"
echo ${arrayOfSwaped[@]}
counter=0
decimalKey=$((2#$allEight))
for j in ${arrayOfSwaped[@]}
do
decimalJ=$((2#$j))
arrayOfXORs[$counter]=$(($decimalJ^$decimalKey))
((counter++))
done

# echo "The XOR text is: " ${arrayOfXORs[@]}
for k in ${arrayOfXORs[@]}
do
   case $k in
    65) echo -n "A" >> decryptionOutput.txt;;
```

Figure 12: Decryption Function

The Output File:

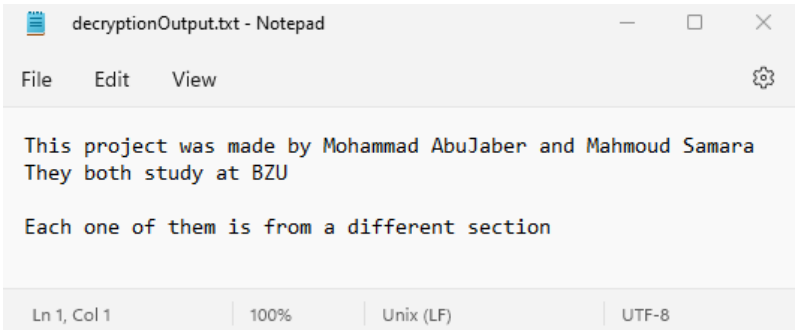

Figure 13: Output Decrypted File

Another Input File:

The Input File:

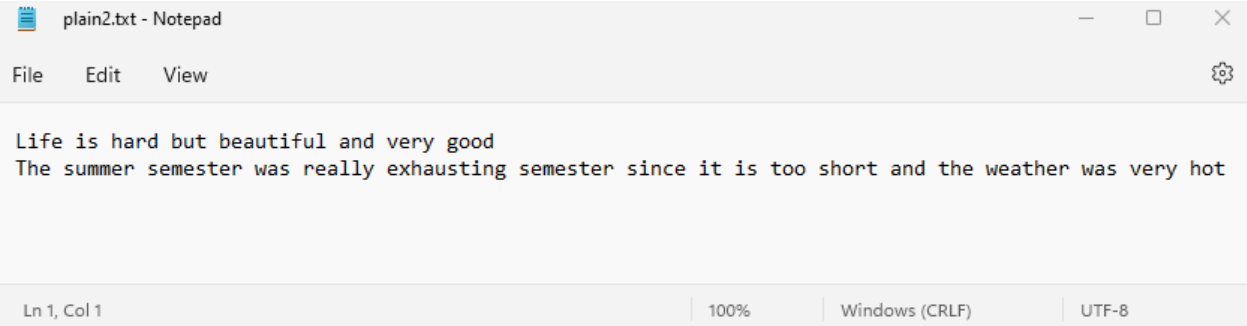

Figure 14: Input File

The Encryption Output:

```
Please enter the letter
e
Please insert the name of plain file:
plain2.txt

--->The file name u entered is exist
All the file data is true and there is no numbers or special characters
The Key In Decimal=  128
The Key In Binary=  10000000
The Encrypted Text=
11001100 10011110 01101110 01011110 11111101 10011110 00111111 11111101 10001110 00011110 00101111 0100
1110 11111101 00101110 01011111 01001111 11111101 00101110 01011110 00011110 01011111 01001111 10011110
  01101110 01011111 11001110 11111101 00011110 11101110 01001110 11111101 01101111 01011110 00101111 10
011111 11111101 01111110 11111110 11111110 01001110 11111101 00001100 01001101 10001110 01011110 111111
01  00111111 01011111 11011110 11011110 01011110 00101111 11111101 00111111 01011110 11011110 01011110
00111111 01001111 01011110 00101111 11111101 01111111 00011110 00111111 11111101 00101111 01011110 0001
1110 11001110 11001110 10011111 11111101 01011110 10001111 10001110 00011110 01011111 00111111 01001111
  10011110 11101110 01111110 11111101 00111111 01011110 11011110 01011110 00111111 01001111 01011110 00
101111 11111101 00111111 10011110 11101110 00111110 01011110 11111101 10011110 01001111 11111101 100111
10  00111111 11111101 01001111 11111110 11111110 11111101 00111111 10001110 11111110 00101111 01001111
11111101 00011110 11101110 01001110 11111101 01001111 10001110 01011110 11111101 01111111 01011110 0001
1110 01001111 10001110 01011110 00101111 11111101 01111111 00011110 00111111 11111101 01101111 01011110
  00101111 10011111 11111101 10001110 11111110 01001111 00001000


The Encrypted text was saved in a file called cipher.txt
```

Figure 15: Encryption Output

The Decryption Output:

```
Please enter the letter
d
Please insert the name of cipher.txt file:
cipher.txt

--->The file name u entered is exist
The key in binary is:  10000000
The key in decimal is:  128


The swaped cipher.txt is:
11001100 11101001 11100110 11100101 11011111 11101001 11110011 11011111 11101000 11100001 11110010 11100100 110111
11 11100010 11110101 11110100 11011111 11100010 11100101 11100001 11110101 11110100 11101001 11100110 11110101 111
01100 11011111 11100001 11101110 11100100 11011111 11110110 11100101 11110010 11111001 11011111 11100111 11101111
11101111 11100100 11011111 11000000 11010100 11101000 11100101 11011111 11110011 11110101 11101101 11101101 111001
01 11110010 11011111 11110011 11100101 11101101 11100101 11110011 11110100 11100101 11110010 11011111 11110111 111
00001 11110011 11011111 11110010 11100101 11100001 11101100 11101100 11111001 11011111 11100101 11111000 11101000
11100001 11110101 11110011 11110100 11101001 11101110 11100111 11011111 11110011 11100101 11101101 11100101 111100
11 11110100 11100101 11110010 11011111 11110011 11101001 11101110 11100011 11100101 11011111 11101001 11110100 110
11111 11101001 11110011 11011111 11110100 11101111 11101111 11011111 11110011 11101000 11101111 11110010 11110100
11011111 11100001 11101110 11100100 11011111 11110100 11101000 11100101 11011111 11110111 11100101 11100001 111101
00 11101000 11100101 11110010 11011111 11110111 11100001 11110011 11011111 11110110 11100101 11110010 11111001 110
11111 11101000 11101111 11110100 10000000


The Decrypted text was saved in a file called decryptionOutput.txt
```

Figure 16: Decryption Output

The decrypted file:

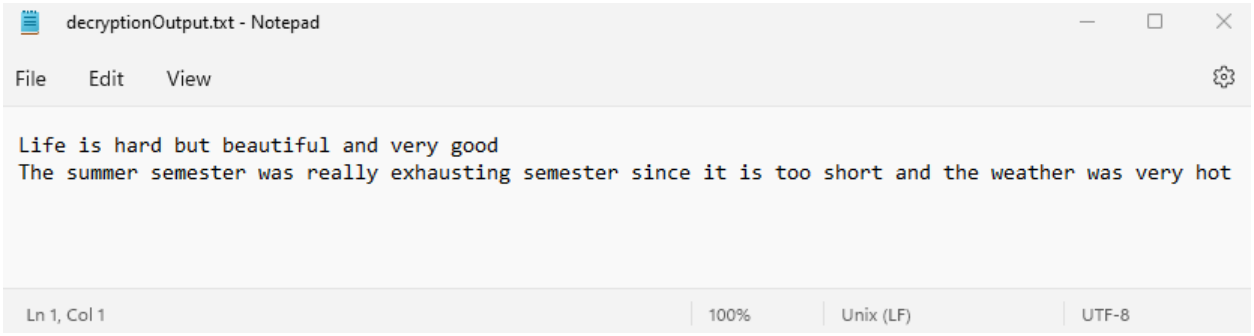

Figure 17: Decrypted File

2. Appendix

```
#!/bin/bash
echo "Welcome to our program"
menu(){
printf "\n\nPlease choose letter to make the specific operation from the following menu\n"
printf "\n\n************** MENU **************\n"
printf "\nE: for Encryption
D: for Decryption
X: Exit\n"
printf "Please enter the letter\n"
}
checkIfFileExist(){
while true # while statement that keep running until user enter an exist file
do
 if [ -e "$file" ] # if statement to check if the file is exist or not
 then
    printf "\n--->The file name u entered is exist\n"
    break
     else
printf "\n--->the file name that u entered is not exist , please enter right file name \n"
read file
   fi # end of if statement that check if file exist or not
done
}
menu
Encryption(){
cat /dev/null > cipher.txt
echo "Please insert the name of plain file:"
read file #reading the file name before entering if statement
checkIfFileExist
charcheck=$(cat $file | grep "[^A-Za-z ]") # check if the file contain any number or not

if [ -n "$charcheck" ]
   then
      printf " Error ==> There is at least a number or special character in your input file
      please check your file and get back to the program\n"
      exit
    else
      printf "All the file data is true and there is no numbers or special characters\n"
fi # end of if statement that check if there is any character
declare -a arrayOfMods
counter=0

letterIndex=0
ALPHABET=(Aa Bb Cc Dd Ee Ff Gg Hh Ii Jj Kk Ll Mm Nn Oo Pp Qq Rr Ss Tt Uu Vv Ww Xx Yy Zz)

for i in $(cat $file | tr " " "\n")
do
```

```
    for j in ${ALPHABET[@]}
    do
        ((++letterIndex))
        if [ "$(echo $i | grep [$j])" ]
        then
            repeat=$(echo $i | tr -cd [$j] | wc -c)
            ((sum+=letterIndex*repeat))
            arrayOfMods[$counter]=$(($sum%256))
        fi
    done
counter=$(($counter+1))
letterIndex=0
sum=0
done

key=${arrayOfMods[0]}
for k in ${arrayOfMods[@]}
do
    if [ $k -gt $key ]
    then
        key=$k
    fi
done
echo "The Key In Decimal= " $key
x=({0..1}{0..1}{0..1}{0..1}{0..1}{0..1}{0..1}{0..1})
binaryKey=${x[$key]}
echo "The Key In Binary= " $binaryKey

##########  XOR with Swap operations ###############

declare -a arrayOfChars
declare -a arrayOfXORs
counter=0
convertToBinary(){
x=({0..1}{0..1}{0..1}{0..1}{0..1}{0..1}{0..1}{0..1})
binaryKey=${x[$tmp]}
arrayOfXORs[$counter]=$binaryKey
counter=$(($counter+1))
}
for i in $(cat $file | tr "\n" "@" | tr " " "_")
    do
    echo $i | sed 's/\(.\)/\1\n/g' >> temp_$$
done
counter=0
for j in $(cat temp_$$)
    do
    arrayOfChars[$counter]=$j
    ((counter++))
done
```

```
rm temp_$$

for k in ${arrayOfChars[@]}
    do
    case $k in
    A)tmp=$((65^$key))
    convertToBinary
    ;;
    B)tmp=$((66^$key))
    convertToBinary
    ;;
    C)tmp=$((67^$key))
    convertToBinary
    ;;
    D)tmp=$((68^$key))
    convertToBinary
    ;;
    E)tmp=$((69^$key))
    convertToBinary
    ;;
    F)tmp=$((70^$key))
    convertToBinary
    ;;
    G)tmp=$((71^$key))
    convertToBinary
    ;;
    H)tmp=$((72^$key))
    convertToBinary
    ;;
    I)tmp=$((73^$key))
    convertToBinary
    ;;
    J)tmp=$((74^$key))
    convertToBinary
    ;;
    K)tmp=$((75^$key))
    convertToBinary
    ;;
    L)tmp=$((76^$key))
    convertToBinary
    ;;
    M)tmp=$((77^$key))
    convertToBinary
    ;;
    N)tmp=$((78^$key))
    convertToBinary
    ;;
    O)tmp=$(( 79^$key))
```

```
    convertToBinary
    ;;
    P)tmp=$((80^$key))
    convertToBinary
    ;;
    Q)tmp=$((81^$key))
    convertToBinary
    ;;
    R)tmp=$((82^$key))
    convertToBinary
    ;;
    S)tmp=$((83^$key))
    convertToBinary
    ;;
    T)tmp=$((84^$key))
    convertToBinary
    ;;
    U)tmp=$((85^$key))
    convertToBinary
    ;;
    V)tmp=$((86^$key))
    convertToBinary
    ;;
    W)tmp=$((87^$key))
    convertToBinary
    ;;
    X)tmp=$((88^$key))
    convertToBinary
    ;;
    Y)tmp=$((89^$key))
    convertToBinary
    ;;
    Z)tmp=$((90^$key))
    convertToBinary
    ;;
    a)tmp=$((97^$key))
    convertToBinary
    ;;
    b)tmp=$((98^$key))
    convertToBinary
    ;;
    c)tmp=$((99^$key))
    convertToBinary
    ;;
    d)tmp=$((100^$key))
    convertToBinary
    ;;
    e)tmp=$((101^$key))
    convertToBinary
```

```
    ;;
    f)tmp=$((102^$key))
    convertToBinary
    ;;
    g)tmp=$((103^$key))
    convertToBinary
    ;;
    h)tmp=$((104^$key))
    convertToBinary
    ;;
    i)tmp=$((105^$key))
    convertToBinary
    ;;
    j)tmp=$((106^$key))
    convertToBinary
    ;;
    k)tmp=$((107^$key))
    convertToBinary
    ;;
    l)tmp=$((108^$key))
    convertToBinary
    ;;
    m)tmp=$((109^$key))
    convertToBinary
    ;;
    n)tmp=$((110^$key))
    convertToBinary
    ;;
    o)tmp=$((111^$key))
    convertToBinary
    ;;
    p)tmp=$((112^$key))
    convertToBinary
    ;;
    q)tmp=$((113^$key))
    convertToBinary
    ;;
    r)tmp=$((114^$key))
    convertToBinary
    ;;
    s)tmp=$((115^$key))
    convertToBinary
    ;;
    t)tmp=$((116^$key))
    convertToBinary
    ;;
    u)tmp=$((117^$key))
    convertToBinary
    ;;
```

```
    v)tmp=$((118^$key))
    convertToBinary
    ;;
    w)tmp=$((119^$key))
    convertToBinary
    ;;
    x)tmp=$((120^$key))
    convertToBinary
    ;;
    y)tmp=$((121^$key))
    convertToBinary
    ;;
    z)tmp=$((122^$key))
    convertToBinary
    ;;
    @)tmp=$((64^$key))
    convertToBinary
    ;;
    _)tmp=$((95^$key))
    convertToBinary
    ;;
    esac
done

echo "The Encrypted Text= "
for l in ${arrayOfXORs[@]}
    do
    firstPart=$(echo -n $l | cut -c1-4)
    secondPart=$(echo -n $l | cut -c5-8)
    echo -n $secondPart$firstPart " "
    echo -n $secondPart$firstPart " " >> cipher.txt
done

x=({0..1}{0..1}{0..1}{0..1}{0..1}{0..1}{0..1}{0..1})
binaryKey=${x[$key]}
firstPart=$(echo -n $binaryKey | cut -c1-4)
secondPart=$(echo -n $binaryKey | cut -c5-8)
echo $secondPart$firstPart
echo -n $secondPart$firstPart " " >> cipher.txt

printf "\n\nThe Encrypted text was saved in a file called cipher.txt"
arrayOfXORs=()
arrayOfMods=()
arrayOfChars=()
menu
################################## End of encryption
      ################################################
}
Decryption(){
```

```
cat /dev/null > decryptionOutput.txt
echo "Please insert the name of cipher.txt file:"
read file #reading the file name before entering if statement
checkIfFileExist
Lasteight=$(awk '{print $NF}' $file) # get the last eight bits of the file
firstPart=$(echo -n $Lasteight | cut -c1-4)
secondPart=$(echo -n $Lasteight | cut -c5-8)
allEight=$secondPart$firstPart
echo "The key in binary is: " $allEight
echo "The key in decimal is: " $((2#$allEight))

declare -a arrayOfSwaped
counter=0
for i in $(cat $file)
do
    firstPart=$(echo -n $i | cut -c1-4)
    secondPart=$(echo -n $i | cut -c5-8)
    allEight=$secondPart$firstPart
    arrayOfSwaped[$counter]=$allEight
    ((counter++))
done
printf "\n\nThe swaped cipher.txt is:\n"
echo ${arrayOfSwaped[@]}
counter=0
decimalKey=$((2#$allEight))
for j in ${arrayOfSwaped[@]}
do
decimalJ=$((2#$j))
arrayOfXORs[$counter]=$(($decimalJ^$decimalKey))
((counter++))
done

# echo "The XOR text is: " ${arrayOfXORs[@]}
for k in ${arrayOfXORs[@]}
do
   case $k in
    65) echo -n "A" >> decryptionOutput.txt;;
    66) echo -n "B" >> decryptionOutput.txt;;
    67) echo -n "C" >> decryptionOutput.txt;;
    68) echo -n "D" >> decryptionOutput.txt;;
    69) echo -n "E" >> decryptionOutput.txt;;
    70) echo -n "F" >> decryptionOutput.txt;;
    71) echo -n "G" >> decryptionOutput.txt;;
    72) echo -n "H" >> decryptionOutput.txt;;
    73) echo -n "I" >> decryptionOutput.txt;;
    74) echo -n "J" >> decryptionOutput.txt;;
    75) echo -n "K" >> decryptionOutput.txt;;
    76) echo -n "L" >> decryptionOutput.txt;;
    77) echo -n "M" >> decryptionOutput.txt;;
```

```
    78) echo -n "N" >> decryptionOutput.txt;;
    79) echo -n "O" >> decryptionOutput.txt;;
    80) echo -n "P" >> decryptionOutput.txt;;
    81) echo -n "Q" >> decryptionOutput.txt;;
    82) echo -n "R" >> decryptionOutput.txt;;
    83) echo -n "S" >> decryptionOutput.txt;;
    84) echo -n "T" >> decryptionOutput.txt;;
    85) echo -n "U" >> decryptionOutput.txt;;
    86) echo -n "V" >> decryptionOutput.txt;;
    87) echo -n "W" >> decryptionOutput.txt;;
    88) echo -n "X" >> decryptionOutput.txt;;
    89) echo -n "Y" >> decryptionOutput.txt;;
    90) echo -n "Z" >> decryptionOutput.txt;;
    97) echo -n "a" >> decryptionOutput.txt;;
    98) echo -n "b" >> decryptionOutput.txt;;
    99) echo -n "c" >> decryptionOutput.txt;;
    100) echo -n "d" >> decryptionOutput.txt;;
    101) echo -n "e" >> decryptionOutput.txt;;
    102) echo -n "f" >> decryptionOutput.txt;;
    103) echo -n "g" >> decryptionOutput.txt;;
    104) echo -n "h" >> decryptionOutput.txt;;
    105) echo -n "i" >> decryptionOutput.txt;;
    106) echo -n "j" >> decryptionOutput.txt;;
    107) echo -n "k" >> decryptionOutput.txt;;
    108) echo -n "l" >> decryptionOutput.txt;;
    109) echo -n "m" >> decryptionOutput.txt;;
    110) echo -n "n" >> decryptionOutput.txt;;
    111) echo -n "o" >> decryptionOutput.txt;;
    112) echo -n "p" >> decryptionOutput.txt;;
    113) echo -n "q" >> decryptionOutput.txt;;
    114) echo -n "r" >> decryptionOutput.txt;;
    115) echo -n "s" >> decryptionOutput.txt;;
    116) echo -n "t" >> decryptionOutput.txt;;
    117) echo -n "u" >> decryptionOutput.txt;;
    118) echo -n "v" >> decryptionOutput.txt;;
    119) echo -n "w" >> decryptionOutput.txt;;
    120) echo -n "x" >> decryptionOutput.txt;;
    121) echo -n "y" >> decryptionOutput.txt;;
    122) echo -n "z" >> decryptionOutput.txt;;
    95) echo -n " "  >> decryptionOutput.txt;;
    64) echo >> decryptionOutput.txt;;
   esac
done
printf "\n\nThe Decrypted text was saved in a file called decryptionOutput.txt"
arrayOfXORs=()
arrayOfSwaped=()
menu
################################## End of decryption
      ################################################
```

```
}

while true #while loop to keep reading the choice until the user enter E
do # beginning of while loop

read enteredletter  # reading the entered choice to know what case to do

case $enteredletter in

E|e)Encryption;;
D|d)Decryption;;
X|x)exit 0;;
*)echo "Please Enter a valid choice from the menu only";;

esac # end of case statement
done
```

All work was done on ubuntu-22.04 operating system using visual studio code (VS Code) as an editor. And all screenshots were taken from us using Git Bash (software) terminal.