Bash Script for Caesar Cipher Encoding and Decoding

Mariam Mourad
20210798
Dina Shweikh
20200565

4/1/2024

—

Operating system security lab

—

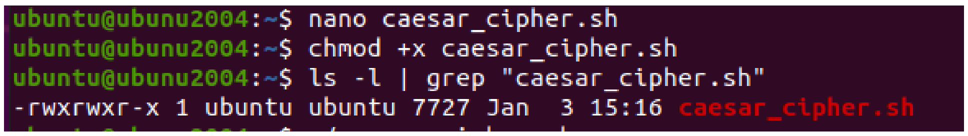
Engineer Hanan Benayan

Objective

The goal of this project is to develop a Bash script that performs both encoding and decoding of text using the Caesar cipher.
The Caesar cipher is a classic substitution cipher where each letter in the plaintext is shifted a certain number of places down or up the alphabet.

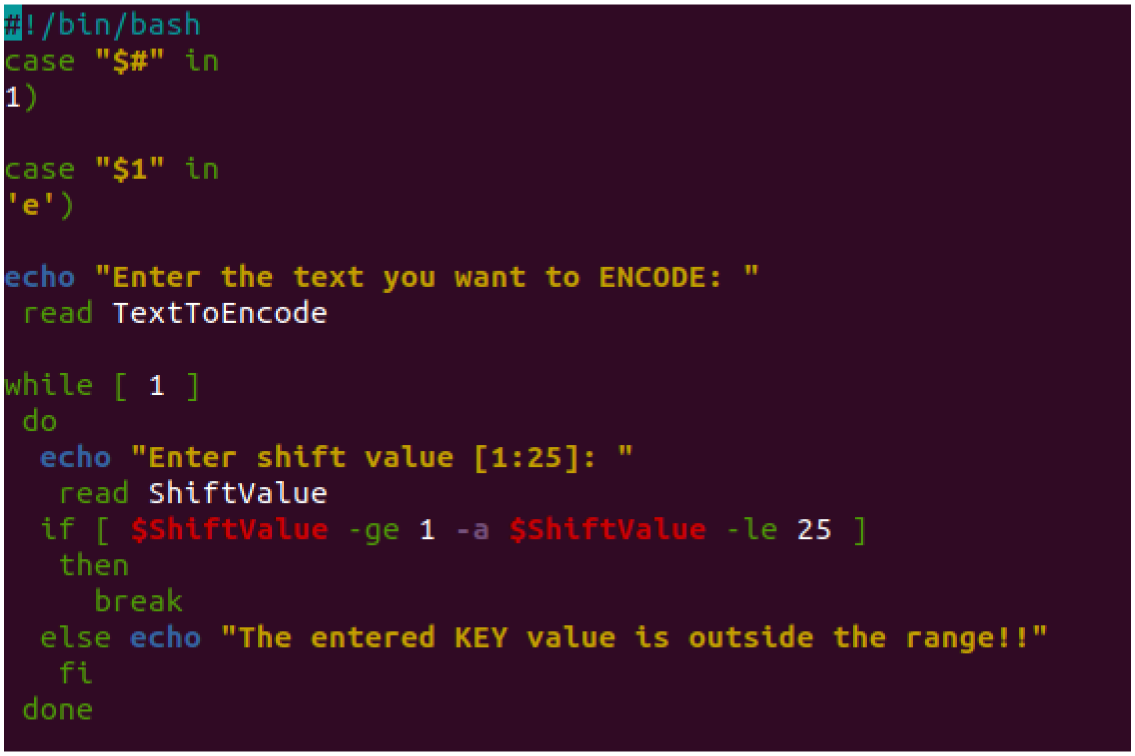
Part 1

First of all we created a file using touch command and changed the mode so we can excute it
Commands used :
Touch typo.sh
Chmod 777 typo.sh



```
ubuntu@ubunu2004:~$ nano caesar_cipher.sh
ubuntu@ubunu2004:~$ chmod +x caesar_cipher.sh
ubuntu@ubunu2004:~$ ls -l | grep "caesar_cipher.sh"
-rwxrwxr-x 1 ubuntu ubuntu 7727 Jan  3 15:16 caesar_cipher.sh
```

Using the command " nano typo.sh " to start writing the script



```
#!/bin/bash
case "$#" in
1)

case "$1" in
'e')

echo "Enter the text you want to ENCODE: "
 read TextToEncode

while [ 1 ]
 do
  echo "Enter shift value [1:25]: "
   read ShiftValue
  if [ $ShiftValue -ge 1 -a $ShiftValue -le 25 ]
   then
     break
  else echo "The entered KEY value is outside the range!!"
   fi
 done
```

Always start the script with #!/bin/bash

Part { 1 }

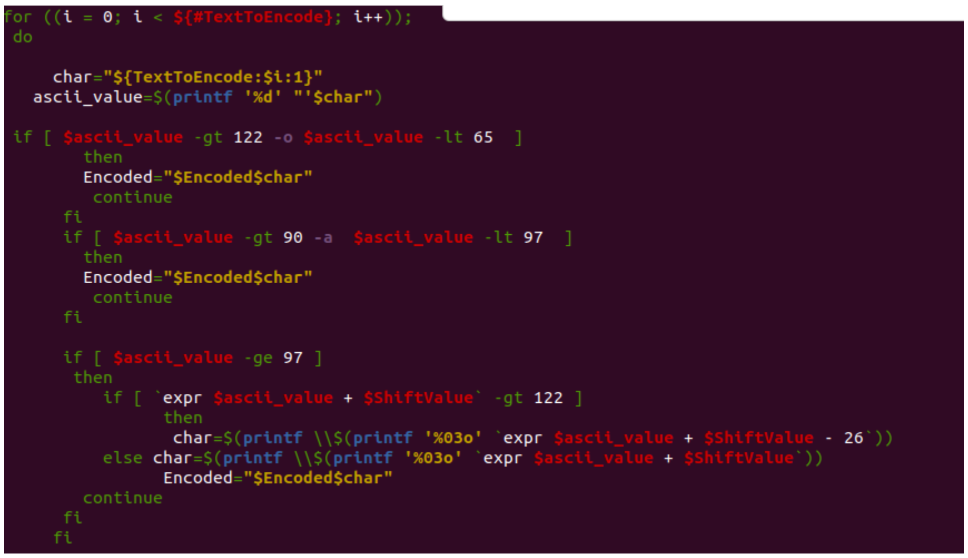
The outer case will be checking on the number of argements sent when the file was excuted; if one argument was sent ->> we will enter an inner case to check if the user wants to Encode || Decode. if two arguments were sent ->> we will enter another inner case that handels characters AND symbolls Encoding/Decoding. Finally if more than two arguments where sent ->> an error msg will appear to the user.

Part 1.2

The first inner loop (one argument e/d):

Case(1): If the user wants to Endoce a text, The user is asked to enter the text and the key value (shiftting offeset).
There's an IF statement to check the validation of the key value (must be between 1 and 25 inclusive), popping an error msg if the entered key violated the condition and ask the user to re-enter the value, it is an infinite loop until a valid key value is entered then we move on to the next step.

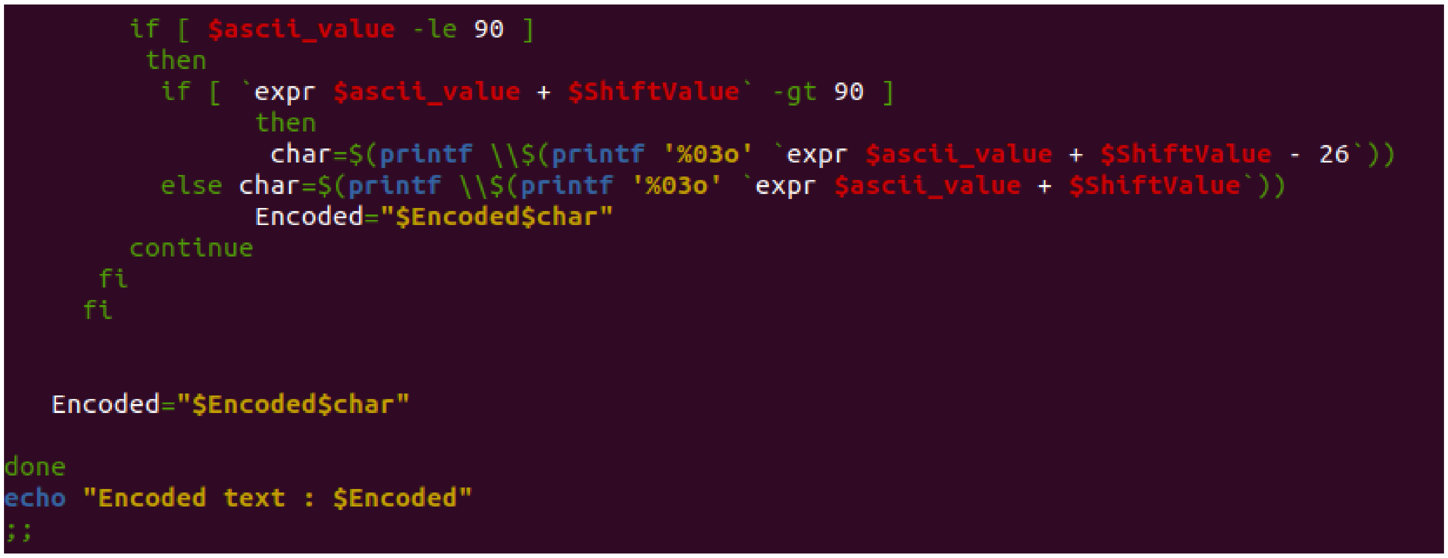
now we want to separate each character to encode it then merge it back to the text, to go through each charater in the text we used a FOR loop (i) scans through the text (${#TextToEncode} gives the length of the text)



```
for ((i = 0; i < ${#TextToEncode}; i++));
 do

    char="${TextToEncode:$i:1}"
  ascii_value=$(printf '%d' "'$char")

 if [ $ascii_value -gt 122 -o $ascii_value -lt 65  ]
       then
       Encoded="$Encoded$char"
        continue
     fi
     if [ $ascii_value -gt 90 -a  $ascii_value -lt 97  ]
       then
       Encoded="$Encoded$char"
        continue
     fi

     if [ $ascii_value -ge 97 ]
      then
         if [ `expr $ascii_value + $ShiftValue` -gt 122 ]
               then
                char=$(printf \\$(printf '%03o' `expr $ascii_value + $ShiftValue - 26`))
         else char=$(printf \\$(printf '%03o' `expr $ascii_value + $ShiftValue`))
               Encoded="$Encoded$char"
       continue
     fi
    fi
```

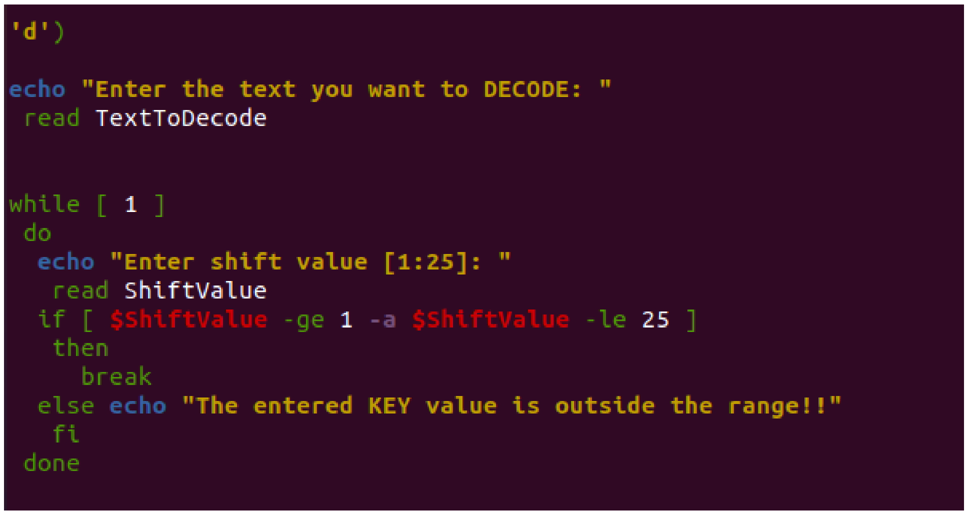
Extracting one character and storing it in Char variable using "${TextToEncode:$i:1}"
then transelating it to its Ascii Code using $(printf '%d' "$char")
Encoded variable will store the encoded text
the first and the second IF statements in the loop will check if Char is a symbol if so it we will merge it as is to Encoded.
The third IF statement will check if Char is a LOWERCASE letter, if so:
We calculate the value of the shifted letter if it exceeds "z" "122 in ascii"
We do a rotation going back to "A". then append the result to Encoded.

This part of the code deals with CAPITAL letters.
The first if condition checks whether intended character to be encoded is capital by checking if its ascii value is greater than 90 or not ($ascii_value –le 90)



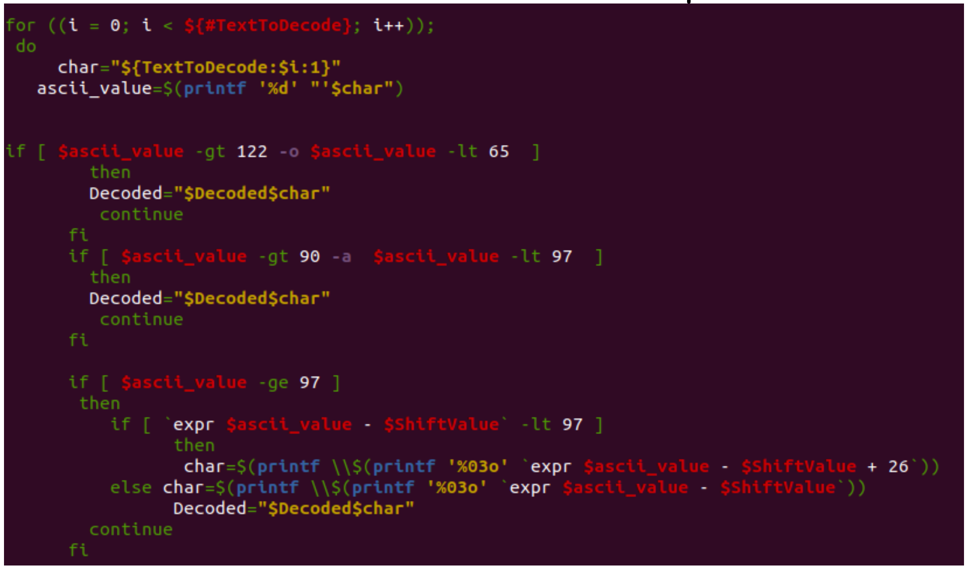
```
        if [ $ascii_value -le 90 ]
         then
          if [ `expr $ascii_value + $ShiftValue` -gt 90 ]
                then
                 char=$(printf \\$(printf '%03o' `expr $ascii_value + $ShiftValue - 26`))
          else char=$(printf \\$(printf '%03o' `expr $ascii_value + $ShiftValue`))
                Encoded="$Encoded$char"
        continue
      fi
     fi


   Encoded="$Encoded$char"

done
echo "Encoded text : $Encoded"
;;
```

$ascii_value is a variable initiated before to calculate the ascii value of any input character.
If true, then another if statement will check if the shifted (Encoded) value of the character will exceed the character 'Z'
('expr $ascii_value + $ShiftValue' -gt 90)
If true, we do a rotation back to 'A' using this command:
Char=$(printf \\$(printf '%03o' expr &ascii_value + $Shiftvalue –26))
The expression '%03o'is used to convert an integer to an octal number. The '0' adds leading zeros and '3' indicates how many digits it will be represented with.
'expr $ascii_value + $Shiftvalue'
Is used to add two string variables as integers.
'-26' will rotate character back to A.
Otherwise, the encoded character is calculated and appended to Encoded.



```
'd')

echo "Enter the text you want to DECODE: "
 read TextToDecode


while [ 1 ]
 do
  echo "Enter shift value [1:25]: "
   read ShiftValue
  if [ $ShiftValue -ge 1 -a $ShiftValue -le 25 ]
   then
     break
  else echo "The entered KEY value is outside the range!!"
   fi
 done
```

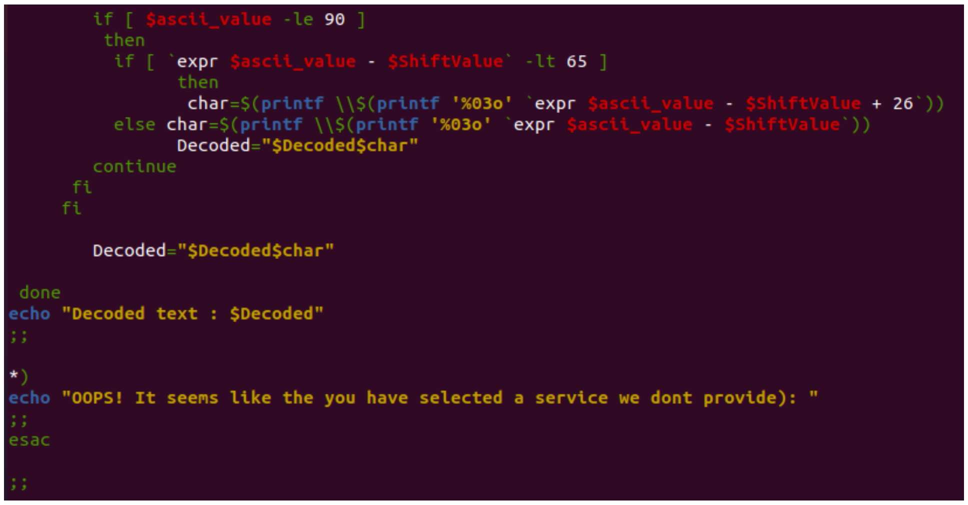
Case(2): If the user wants to Decode a text, The user is asked to enter the text and the key value (shiftting offeset) to return the original string entered before it was encoded.
There's an IF statement to check the validation of the key value (must be between 1 and 25 inclusive), popping an error msg if the entered key violated the condition and ask the user to re-enter the value, it is an infinite loop until a valid key value is entered then we move on to the next step.



```
for ((i = 0; i < ${#TextToDecode}; i++));
 do
     char="${TextToDecode:$i:1}"
   ascii_value=$(printf '%d' "'$char")


if [ $ascii_value -gt 122 -o $ascii_value -lt 65  ]
        then
        Decoded="$Decoded$char"
         continue
      fi
      if [ $ascii_value -gt 90 -a  $ascii_value -lt 97  ]
        then
        Decoded="$Decoded$char"
         continue
      fi

      if [ $ascii_value -ge 97 ]
       then
          if [ `expr $ascii_value - $ShiftValue` -lt 97 ]
                then
                 char=$(printf \\$(printf '%03o' `expr $ascii_value - $ShiftValue + 26`))
          else char=$(printf \\$(printf '%03o' `expr $ascii_value - $ShiftValue`))
                Decoded="$Decoded$char"
        continue
      fi
```

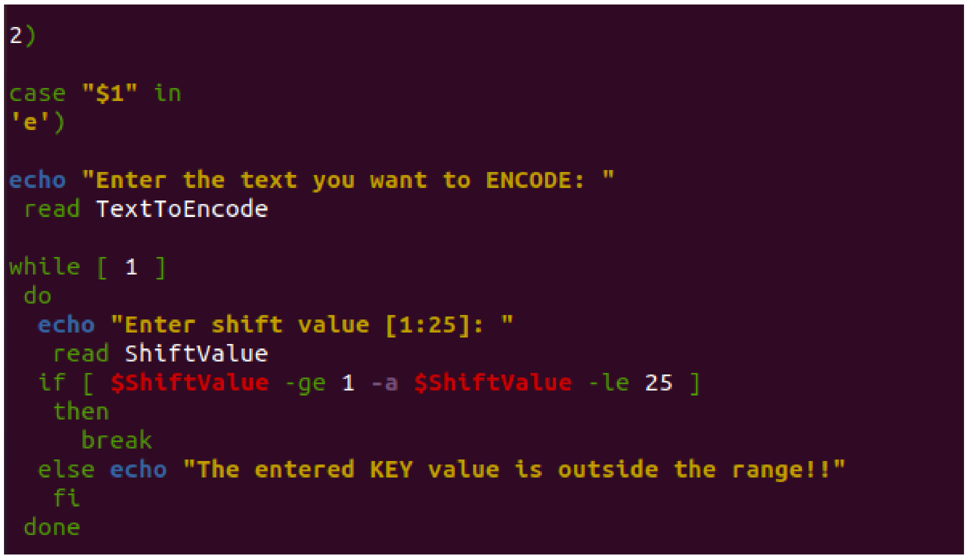
Next, this for loop will check each character entered (${#TextToDecode})
Each extracted character will be stored in the variable char using this command:
Char="&{TextToDecode:&i:1}"

Similar to the previous process, the ascii value of each character is calculated using this command:
Ascii_value=$(printf '%d' "$char")
The following step are two if conditions that checks whether the intended character to be decoded is a symbol or not, if yes, it will be appended to the variable Decoded.
The next nested if statement will check if the character is a LOWER CASE character, if yes, the next if statement will check if the character is CAPITAL after being decoded (subtract shift value from ascii code), then it will decode the character and return it as a LOWER case letter again (+26).
If not, the character will be docoded normally and stored in the variable "char".
Finally, char will be appended to Decoded.



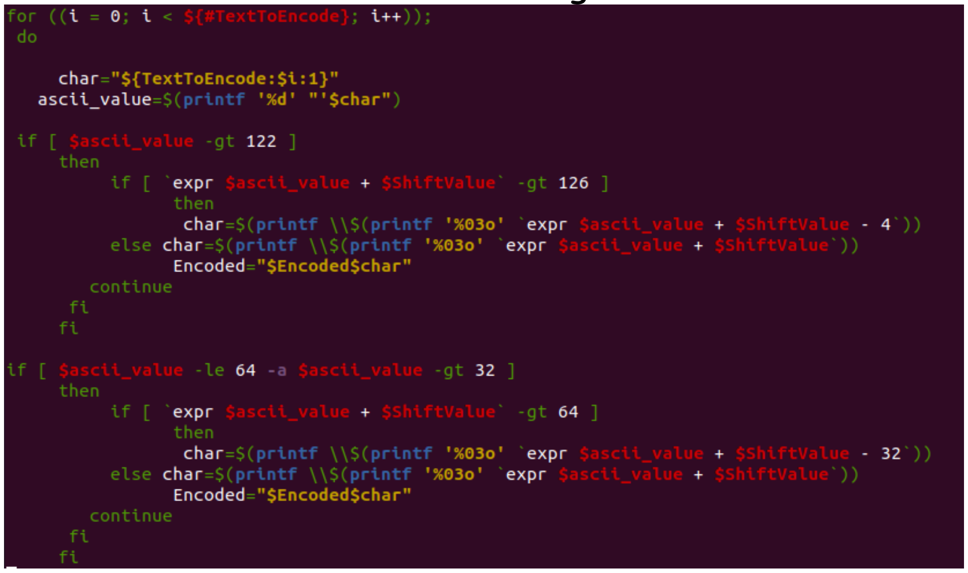
```
        if [ $ascii_value -le 90 ]
         then
          if [ `expr $ascii_value - $ShiftValue` -lt 65 ]
                then
                 char=$(printf \\$(printf '%03o' `expr $ascii_value - $ShiftValue + 26`))
          else char=$(printf \\$(printf '%03o' `expr $ascii_value - $ShiftValue`))
                Decoded="$Decoded$char"
        continue
      fi
     fi

        Decoded="$Decoded$char"

 done
echo "Decoded text : $Decoded"
;;

*)
echo "OOPS! It seems like the you have selected a service we dont provide): "
;;
esac

;;
```

The decoded string will be printed.
If the user entered more than one argument a statment will be displayed "OOPS!, It seems like you have selected a service we dont provide):"



```
2)

case "$1" in
'e')

echo "Enter the text you want to ENCODE: "
 read TextToEncode

while [ 1 ]
 do
  echo "Enter shift value [1:25]: "
   read ShiftValue
  if [ $ShiftValue -ge 1 -a $ShiftValue -le 25 ]
   then
     break
  else echo "The entered KEY value is outside the range!!"
   fi
 done
```

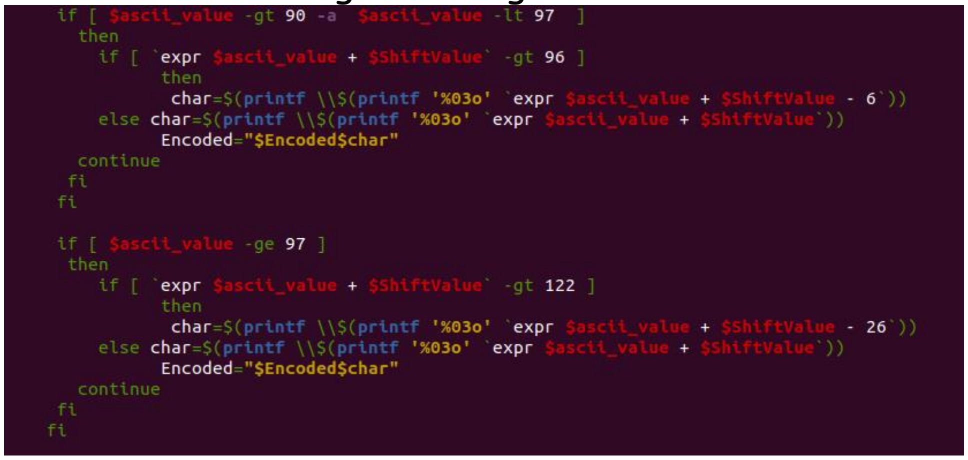
The next case deals with the user sending two arguments.
The code reads the intended string to be decoded ,storing it in the variable "TextToEncode"
An inifinte loop is then initiated to check wether the entered shift value is in the range [1 – 25] or not, if not a message is desplayed "The entered KEY value is outside the range! !".



```
for ((i = 0; i < ${#TextToEncode}; i++));
 do

     char="${TextToEncode:$i:1}"
   ascii_value=$(printf '%d' "'$char")

 if [ $ascii_value -gt 122 ]
     then
          if [ `expr $ascii_value + $ShiftValue` -gt 126 ]
                then
                 char=$(printf \\$(printf '%03o' `expr $ascii_value + $ShiftValue - 4`))
          else char=$(printf \\$(printf '%03o' `expr $ascii_value + $ShiftValue`))
                Encoded="$Encoded$char"
        continue
      fi
     fi

if [ $ascii_value -le 64 -a $ascii_value -gt 32 ]
     then
          if [ `expr $ascii_value + $ShiftValue` -gt 64 ]
                then
                 char=$(printf \\$(printf '%03o' `expr $ascii_value + $ShiftValue - 32`))
          else char=$(printf \\$(printf '%03o' `expr $ascii_value + $ShiftValue`))
                Encoded="$Encoded$char"
        continue
      fi
     fi
```

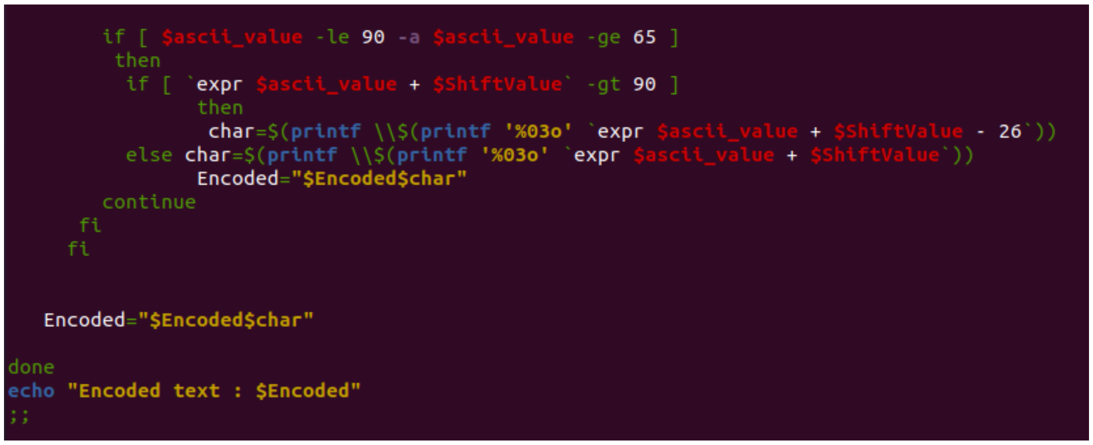
The initial if statment will scan each symbol entered.
Each symbol will be extracted and stored in the variable "char"
The next if statement will check if the character is a special character.
If so, the next if statement will check if the encoded character's ascii code is greater than 126. This can be due to a user entering a large shift value.
So the encoded symbol will be rotated back to 123-126.

Else, the symbol is encoded normally by adding the ascii value to the shift value and appending it to Encoded.
The third if statement will check if the symbol if between 32 and 64, which is the ascii code range of special characters, and again, if the encoded symbol exceeded 64 due to a large shift number, then it is rotated back to the special character's range starting from 32.



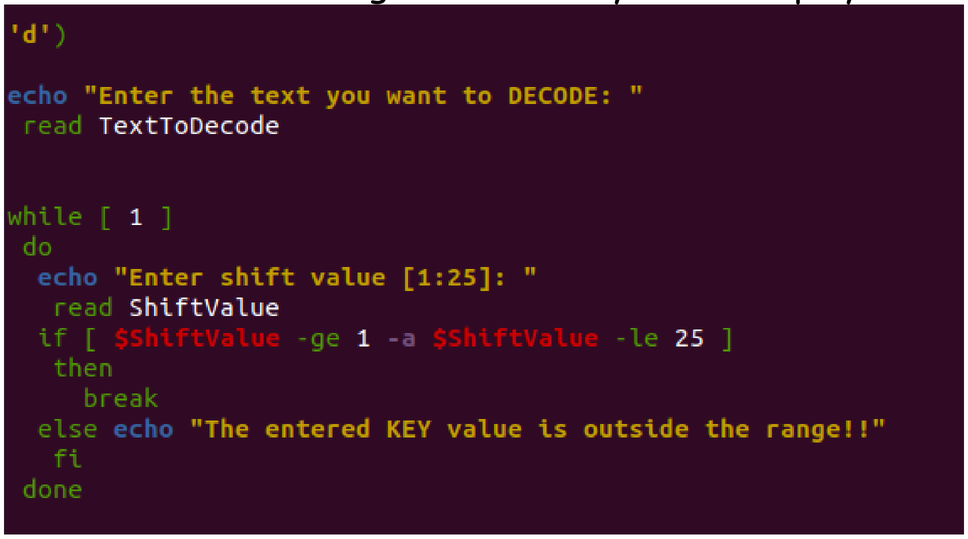
```
        if [ $ascii_value -gt 90 -a  $ascii_value -lt 97  ]
          then
            if [ `expr $ascii_value + $ShiftValue` -gt 96 ]
                  then
                   char=$(printf \\$(printf '%03o' `expr $ascii_value + $ShiftValue - 6`))
            else char=$(printf \\$(printf '%03o' `expr $ascii_value + $ShiftValue`))
                  Encoded="$Encoded$char"
          continue
         fi
        fi

        if [ $ascii_value -ge 97 ]
         then
            if [ `expr $ascii_value + $ShiftValue` -gt 122 ]
                  then
                   char=$(printf \\$(printf '%03o' `expr $ascii_value + $ShiftValue - 26`))
            else char=$(printf \\$(printf '%03o' `expr $ascii_value + $ShiftValue`))
                  Encoded="$Encoded$char"
          continue
        fi
       fi
```

Special characters can also exist between this ascii code range 90-97
So if the encoded symbol is greater than 96 the symbol will be rotated back.
The next if statement will check if the ascii value is greater than 97, which is the first ascii code of small letters 'a'. If so, the next if statement will check whether the encoded symbol's ascii is greater than 122. To check if it lies in the range 123-126 which is the range of other special characters. If so, 26 will be subtracted from the ascii value to rotate the letter back to 'a' or more depending on the ascii value.



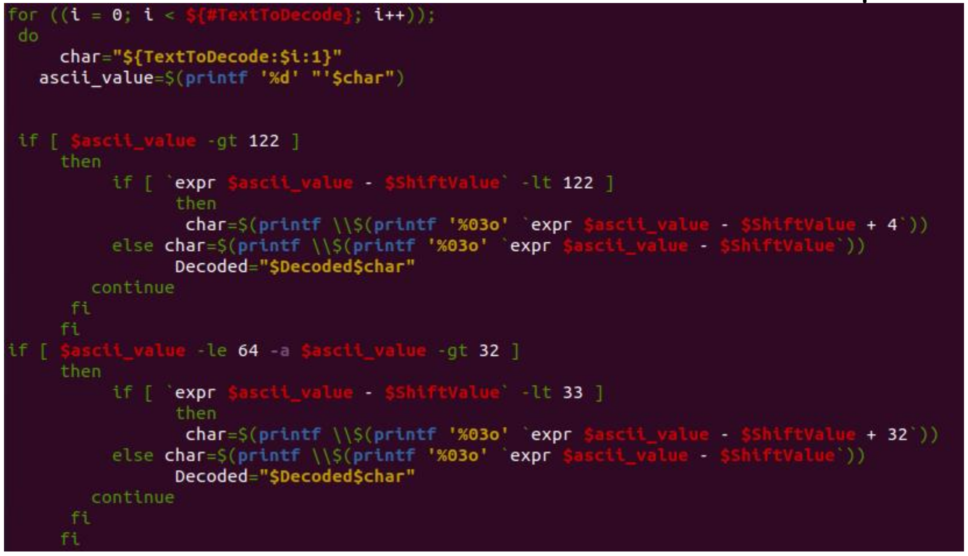
```
        if [ $ascii_value -le 90 -a $ascii_value -ge 65 ]
         then
          if [ `expr $ascii_value + $ShiftValue` -gt 90 ]
                then
                 char=$(printf \\$(printf '%03o' `expr $ascii_value + $ShiftValue - 26`))
          else char=$(printf \\$(printf '%03o' `expr $ascii_value + $ShiftValue`))
                Encoded="$Encoded$char"
        continue
      fi
     fi


   Encoded="$Encoded$char"

done
echo "Encoded text : $Encoded"
;;
```

The following nested if will check if the shifted encoded symbol is CAPITAL. If so, the next if statement will check if the encoded capital letter became a symbol and will rotate it back to 65 (or more depending on ascii value).
The encoded string is then ready to be displayed.



```
'd')

echo "Enter the text you want to DECODE: "
 read TextToDecode


while [ 1 ]
 do
  echo "Enter shift value [1:25]: "
   read ShiftValue
  if [ $ShiftValue -ge 1 -a $ShiftValue -le 25 ]
   then
     break
  else echo "The entered KEY value is outside the range!!"
   fi
 done
```

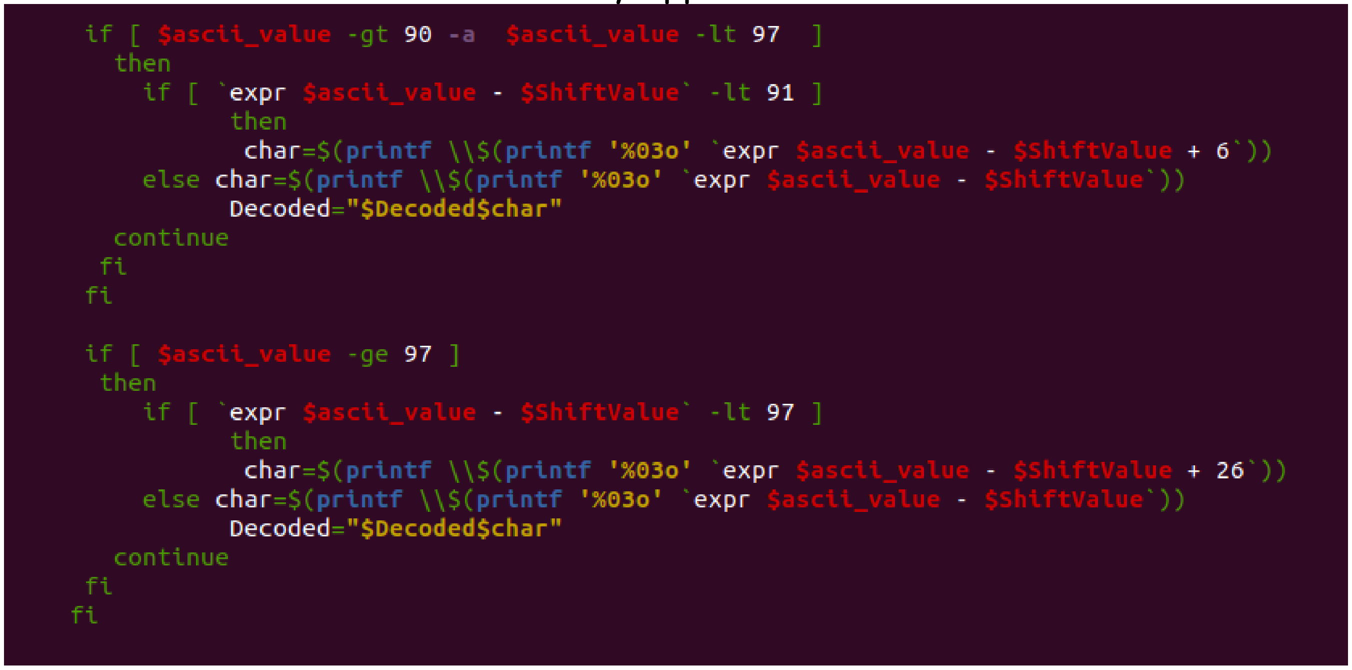
The last case deals with the user entering two arguments. If the first argument entered is 'd', then the user intends to decode the string or character.
The input string will be stored in the variable 'TextToDecode'.
An inifinte loop will be initiated to read and check the input shift value.
If it was outside the range 1-25 , an error message will be popped, allowing the user to re enter the shift value.
If not, the 'break' statement will exit the loop.



```
for ((i = 0; i < ${#TextToDecode}; i++));
 do
     char="${TextToDecode:$i:1}"
   ascii_value=$(printf '%d' "'$char")


 if [ $ascii_value -gt 122 ]
     then
          if [ `expr $ascii_value - $ShiftValue` -lt 122 ]
                then
                 char=$(printf \\$(printf '%03o' `expr $ascii_value - $ShiftValue + 4`))
          else char=$(printf \\$(printf '%03o' `expr $ascii_value - $ShiftValue`))
                Decoded="$Decoded$char"
        continue
      fi
     fi
if [ $ascii_value -le 64 -a $ascii_value -gt 32 ]
     then
          if [ `expr $ascii_value - $ShiftValue` -lt 33 ]
                then
                 char=$(printf \\$(printf '%03o' `expr $ascii_value - $ShiftValue + 32`))
          else char=$(printf \\$(printf '%03o' `expr $ascii_value - $ShiftValue`))
                Decoded="$Decoded$char"
        continue
      fi
     fi
```

The for loop will scan the entered characters.

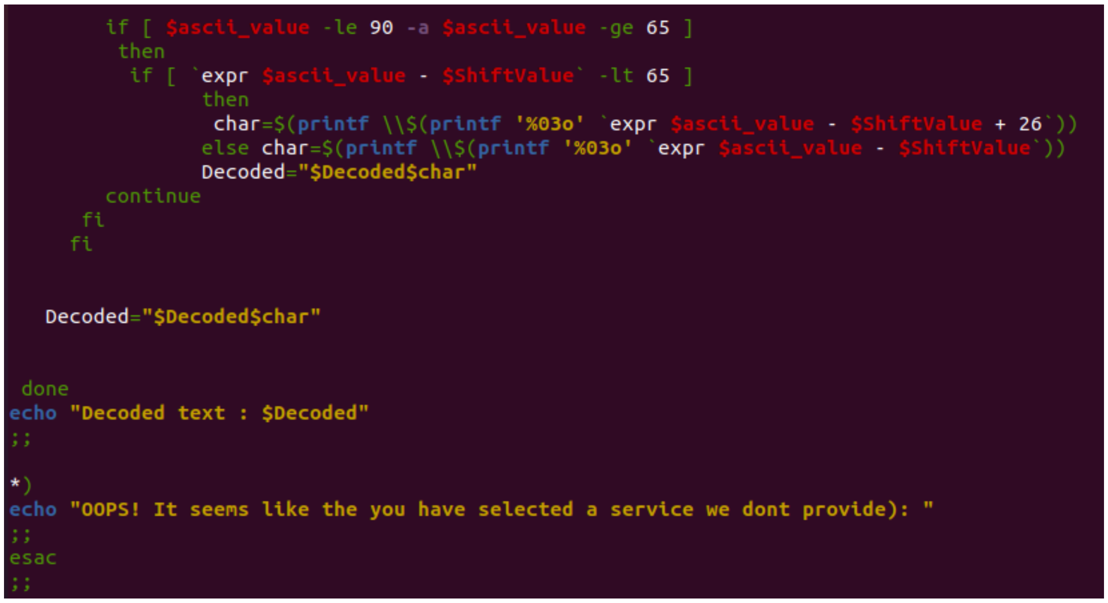
Each character will be extracted and stored in the variable "char", as well as, calculating its ascii value.
The next if statement will check if the calculated ascii value is greater than 122 or not. This will make sure that the ascii code is a special character since special charcters lie in the range [123-126].
If so, 4 will be added to the decoded character to return the character back to the range [123-126].
As mentioned earlier, special characters can lie in the range 33-64. So the next if statement will check
If the ascii code is between this range. If true, the next if statement will check if the decoded symbol is less than 33, which is outside the range of special characters. If so, 32 will be added to the shifted ascii code which will return the ascii value back to the range.
Otherwise, the symbol is decoded normally by subtracting the shift value from the ascii value.
Char is eventually appended to Decoded.



```
     if [ $ascii_value -gt 90 -a  $ascii_value -lt 97  ]
       then
         if [ `expr $ascii_value - $ShiftValue` -lt 91 ]
               then
                char=$(printf \\$(printf '%03o' `expr $ascii_value - $ShiftValue + 6`))
         else char=$(printf \\$(printf '%03o' `expr $ascii_value - $ShiftValue`))
               Decoded="$Decoded$char"
       continue
      fi
     fi

     if [ $ascii_value -ge 97 ]
      then
         if [ `expr $ascii_value - $ShiftValue` -lt 97 ]
               then
                char=$(printf \\$(printf '%03o' `expr $ascii_value - $ShiftValue + 26`))
         else char=$(printf \\$(printf '%03o' `expr $ascii_value - $ShiftValue`))
               Decoded="$Decoded$char"
       continue
     fi
    fi
```

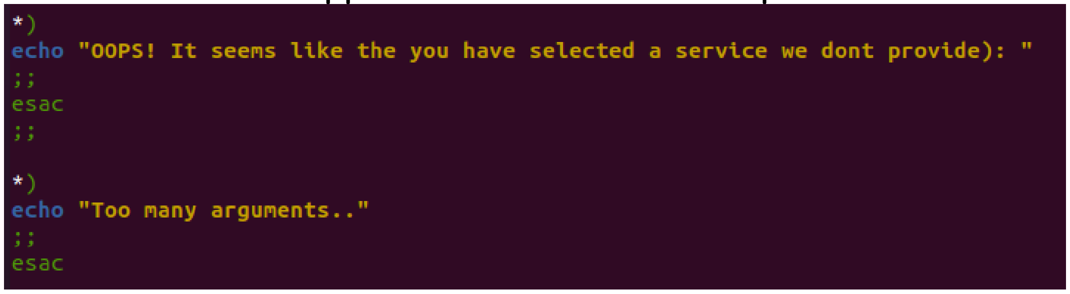
Special character's ascii code can also lie in the range 91-96, so the first if statement will check if the ascii code lies in this range, if it does, another if statement will check if the decoded symbol's ascii is less than 91. If true, this means that the ascii code is outside the range of special characters and is now in the range of LOWERcase letters.

Adding 6 to the shifted ascii code will return it to the range 91-96.
The next if statement will check if the ascii code is greater than or equal to 97, which is the first small character's ascii code 'a'. If this statement was true, then another if statement checks if the shifted ascii code is less than 97 which is a symbol and adds 26 to return it to capital letter's range >97.
Otherwise, the symbol is decoded normally by subtracting the shift value from it's ascii code.



```
       if [ $ascii_value -le 90 -a $ascii_value -ge 65 ]
        then
         if [ `expr $ascii_value - $ShiftValue` -lt 65 ]
                then
                 char=$(printf \\$(printf '%03o' `expr $ascii_value - $ShiftValue + 26`))
                else char=$(printf \\$(printf '%03o' `expr $ascii_value - $ShiftValue`))
                Decoded="$Decoded$char"
       continue
     fi
    fi


  Decoded="$Decoded$char"


 done
echo "Decoded text : $Decoded"
;;

*)
echo "OOPS! It seems like the you have selected a service we dont provide): "
;;
esac
;;
```

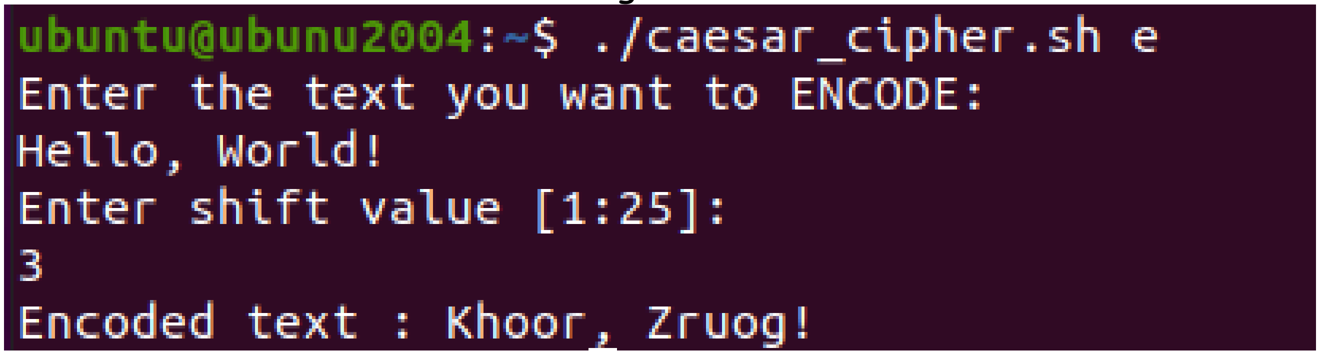
Finally, the next if statement checks if the symbol is a small character, if so, another if statement will check if the shifted ascii code is less than 65 which is outside the range of lowercase characters. 26 will be added to the shifted ascii code to rotate it back to the range 'A-Z'. For example, if the input character is 'A' and the shift value is 2, then the decoded character will become '?'. So adding 26 will return it to the range and will become the letter 'Y'. Otherwise, the symbol is decoded normally by subtracting the shift value from the ascii code.
Char is appended to Decoded and printed.



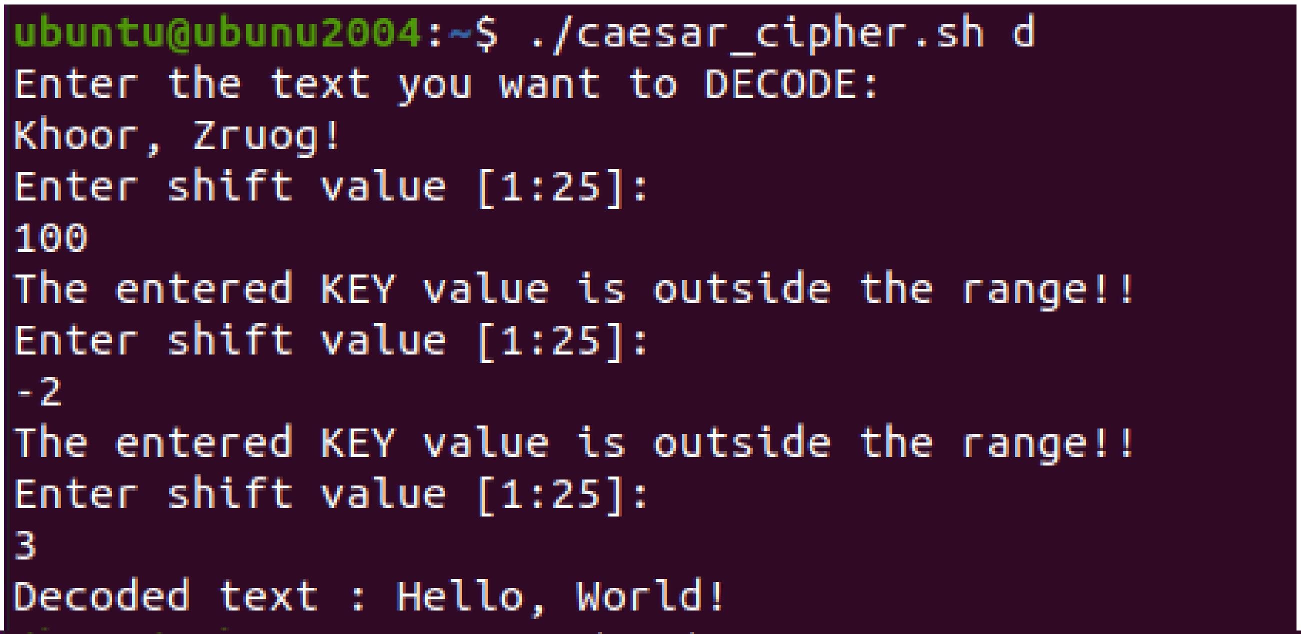
```
*)
echo "OOPS! It seems like the you have selected a service we dont provide): "
;;
esac
;;

*)
echo "Too many arguments.."
;;
esac
```

If the user enters more than two arguments, an error message will be popped "Too many arguments.."

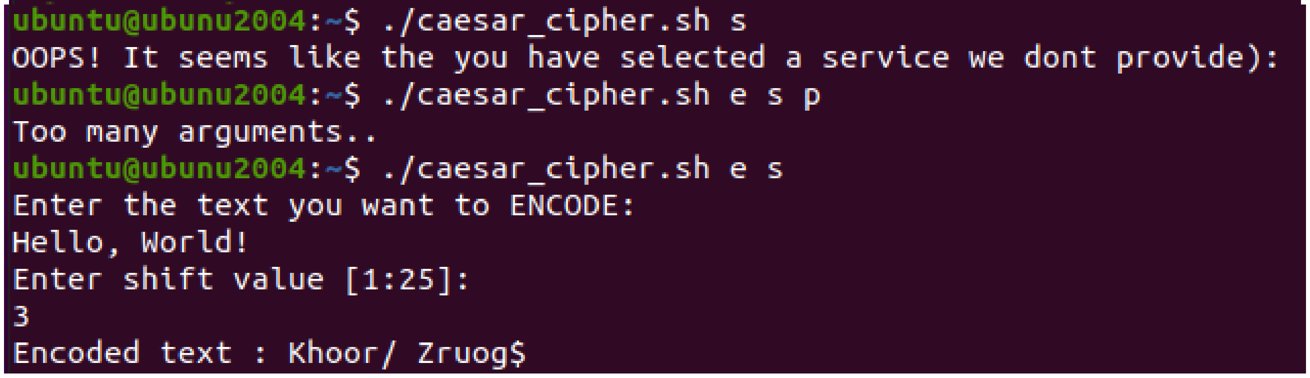
Testing the code:



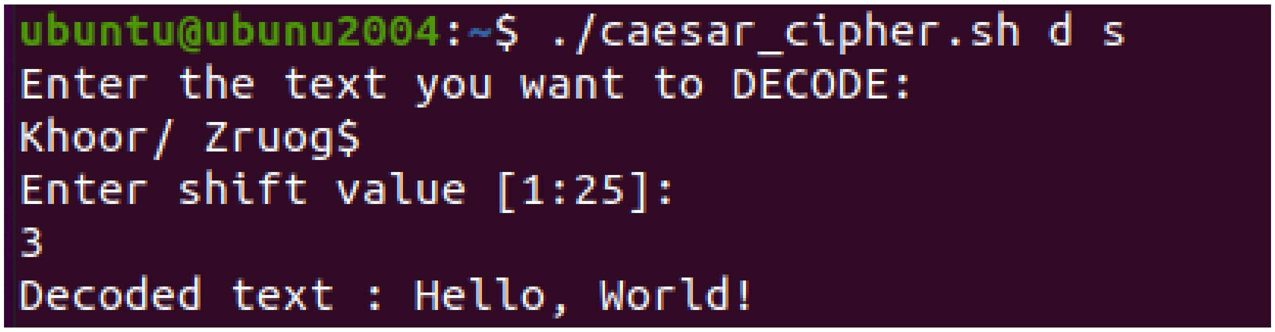
```
ubuntu@ubunu2004:~$ ./caesar_cipher.sh e
Enter the text you want to ENCODE:
Hello, World!
Enter shift value [1:25]:
3
Encoded text : Khoor, Zruog!
```



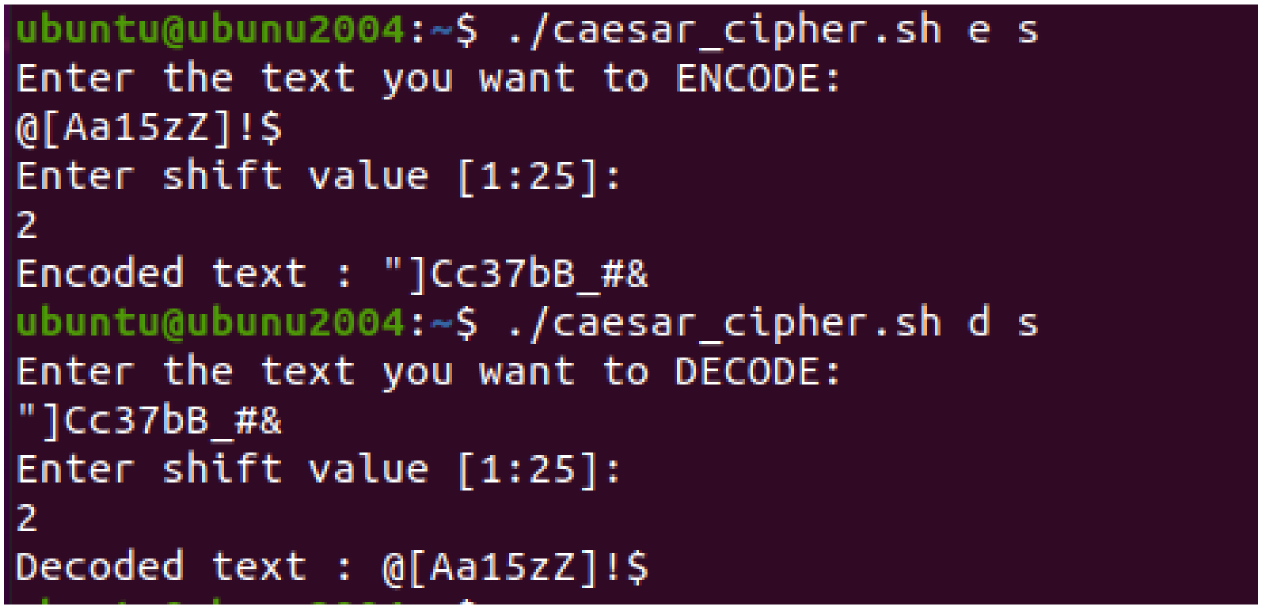
```
ubuntu@ubunu2004:~$ ./caesar_cipher.sh d
Enter the text you want to DECODE:
Khoor, Zruog!
Enter shift value [1:25]:
100
The entered KEY value is outside the range!!
Enter shift value [1:25]:
-2
The entered KEY value is outside the range!!
Enter shift value [1:25]:
3
Decoded text : Hello, World!
```



```
ubuntu@ubunu2004:~$ ./caesar_cipher.sh s
OOPS! It seems like the you have selected a service we dont provide):
ubuntu@ubunu2004:~$ ./caesar_cipher.sh e s p
Too many arguments..
ubuntu@ubunu2004:~$ ./caesar_cipher.sh e s
Enter the text you want to ENCODE:
Hello, World!
Enter shift value [1:25]:
3
Encoded text : Khoor/ Zruog$
```



```
ubuntu@ubunu2004:~$ ./caesar_cipher.sh d s
Enter the text you want to DECODE:
Khoor/ Zruog$
Enter shift value [1:25]:
3
Decoded text : Hello, World!
```



```
ubuntu@ubunu2004:~$ ./caesar_cipher.sh e s
Enter the text you want to ENCODE:
@[Aa15zZ]!$
Enter shift value [1:25]:
2
Encoded text : "]Cc37bB_#&
ubuntu@ubunu2004:~$ ./caesar_cipher.sh d s
Enter the text you want to DECODE:
"]Cc37bB_#&
Enter shift value [1:25]:
2
Decoded text : @[Aa15zZ]!$
```