REPORT

Introduction to Linux and Shell Scripting

Optional Elective

2023-2024*

Group 2

Kanak Agarwal[1,†], Pranav Karkera[1,‡], Chirag Kumar Jain[1,§]
Shreyas V S[1,‖], Tushar Raj [2,#]

[1] Department of Aeronautical and Automobile Engineering, Manipal Institute of Technology, Manipal, Karnataka, India - 576104
[2] Department of Electronics and Instrumentation Engineering, Manipal Institute of Technology, Manipal, Karnataka, India - 576104
[*] Report Submitted on the 9^{th} of April, 2024

[†] Registration Number - 210933058
[‡] Registration Number - 210933046
[§] Registration Number - 210933062
[‖] Registration Number - 210933028
[#] Registration Number - 210932196

1 Introduction

A *Power Calculator* was developed according to the problem statement. It has the following modules:

Alpha Power:

It accepts a word (7-9 characters) and a number. It then displays the word with the middle letter repeated as many times as the number entered. A detailed log of each operation is stored in the file *Alpha* (.txt file).

Expo Power:

This module tests your math skills! It accepts a base number (positive integer between 2 and 6) and an exponent (both positive and negative). It then compares your guess with the actual answer. A detailed log of each attempt is stored in the file *LiPowerFile* (.txt file).

2 Code

```
#!/bin/bash
# Optional Elective – Introduction to Linux and Shell Scripting
# Assignment/FISAC
# Academic Year 2023–24
# Batch of 2021–25
# Group 2
# Code Developed on April 8th 2024

# Pre Processing
touch Alpha.txt
touch LiPowerFile.txt

# Header Write
echo "Word:Number:Modified Word" >> Alpha.txt
echo "S_No:Base Number:Exponent:Guess:Answer" >> LiPowerFile.txt

# Banner
echo ""
echo -e "************************************************** \e[21;1mUnleash The Power Calculator\e[0m **************************************************"
```

```
20  echo ""
21
22  echo −e "\e[5mWelcome to the \e[1mPower Calculator!\e[0m"
23
24  echo ""
25
26  echo −e "\e[1m\e[4:3mUsage Notes\e[0m"
27  echo ""
28  echo −e "The \e[3mPower Calculator\e[0m has the following submodules:"
29  echo ""
30  echo −e "\e[4mAlpha Power:\e[0m"
31  echo ""
32  echo −e "It accepts a word (7−9 characters) and a number. It then displays the word with the middle letter repeated as many times as the number entered.
33  A detailed log of each operation is stored in the file \e[3mAlpha\e[0m (.txt file)."
34
35  echo ""
36
37  echo −e "\e[4mExpo Power:\e[0m"
38  echo ""
39  echo −e "This module tests your math skills! It accepts a base number (positive integer between 2 and 6) and an exponent (both positive and negative).
40  It then compares your guess with the actual answer. A detailed log of each attempt is stored in the file \e[3mLiPowerFile\e[0m (.txt file)."
41
42  echo ""
43
44  echo "**********************************************************************************************************************************************"
45  echo ""
46
47  # Actual Code
48
49  while true # Outer While controls the main menu
50  do
51    echo −e "\e[5mWelcome to the \e[1mPower Calculator!\e[0m"
52    echo ""
53    echo −e "\e[1m\e[4:3mMain Menu\e[0m"
54    echo ""
55    echo −e "\e[3mChoice\e[0m − Operation"
56    echo ""
57    echo −e "\e[3mman\e[0m − Display Usage Notes"
58    echo −e "\e[3mAP\e[0m − Launch Alpha Power"
```

```
    echo -e "\e[3mEP\e[0m - Launch Expo Power"
    echo -e "\e[3m9\e[0m - Exit the Calculator"
    echo ""
    read -p "Which operation would you like to do? " c
    case $(tr '[:upper:]' '[:lower:]' <<< "$c") in # global case - esac construct
      "man")
      echo ""
      echo "*****************************************************************************************************************************************"
      echo -e "\e[1m\e[4:3mUsage Notes\e[0m"
      echo ""
      echo -e "The \e[3mPower Calculator\e[0m has the following submodules:"
      echo ""
      echo -e "\e[4mAlpha Power:\e[0m"
      echo ""
      echo -e "It accepts a word (7-9 characters) and number. It then displays the word with the middle letter repeated as many times as the number entered.
      A detailed log of each operation is stored in the file \e[3mAlpha\e[0m (.txt file)."

      echo ""

      echo -e "\e[4mExpo Power:\e[0m"
      echo ""
      echo -e "This module tests your math skills! It accepts a base number (positive integer between 2 and 6) and an exponent (both positive and negative).
      It then compares your guess with the actual answer. A detailed log of each attempt is stored in the file \e[3mLiPowerFile\e[0m (.txt file)."
      echo "*****************************************************************************************************************************************"

      echo ""
      ;;

      "ap") # Alpha Power Case
          while true # Inner While Controls the local menu
          do
            echo ""
            echo "*****************************************************************************************************************************************"
            echo ""
            echo -e "\e[5mWelcome to the \e[1mAlpha Power Module!\e[0m"
            echo ""
            echo -e "\e[3mChoice\e[0m - Operation"
            echo ""
            echo -e "\e[3mcalc\e[0m - Use the Calculator"
```

```
98      echo -e "\e[3mlog\e[0m - View the Log file"
99      echo -e "\e[3mres\e[0m - Reset the Log file"
100     echo -e "\e[3m9\e[0m - Exit the module"
101     echo ""
102     read -p "Which operation would you like to do? " ch
103     case $(tr '[:upper:]' '[:lower:]' <<< "$ch") in # local case - esac construct
104       "calc")
105           i='y'
106           while true
107           do
108             echo ""
109             read -p "Enter the word to be operated on (7-9 characters only): " word
110             len=${#word}
111             if [ -z $word ]; then # empty condition
112                     echo "Word is empty, try again!"
113                     continue
114             fi
115             if ! (( $len >= 7 && $len <= 9 )); then # length range condition
116               echo ""
117               echo "The word should be only 7-9 characters long!"
118               read -p "Do you want to try again (y/n)? " i
119               if [ $(tr '[:upper:]' '[:lower:]' <<< "$i") == 'y' ]; then
120                 continue
121               else
122                 break
123               fi
124             elif [[ $word =~ [^[:alpha:]] ]]; then # alphabetical condition
125               echo ""
126               echo "The word should contain only alphabets!"
127               read -p "Do you want to try again (y/n)? " i
128               if [ $(tr '[:upper:]' '[:lower:]' <<< "$i") == 'y' ]; then
129                 continue
130               else
131                 break
132               fi
133             else
134               read -p "Enter the number of times the middle letter is to be displayed: " num
135               if [ -z $num ]; then # empty condition
136                       echo "Number is empty, try again!"
```

```
            continue
        fi
        if [[ ! $num =~ [0-9]+ ]]; then # numerical condition
            echo ""
            echo "The input should contain only numbers!"
            read -p "Do you want to try again (y/n)? " i
            if [ $(tr '[:upper:]' '[:lower:]' <<< "$i") == 'y' ]; then
                continue
            else
                break
            fi
        fi
        if [[ $num =~ [0-9]+.[0-9]+ ]]; then # integer condition
            echo ""
            echo "The input should be an integer!"
            read -p "Do you want to try again (y/n)? " i
            if [ $(tr '[:upper:]' '[:lower:]' <<< "$i") == 'y' ]; then
                continue
            else
                break
            fi
        fi

        if (( ${#word} % 2 == 0 )); then # Calculate the middle letter(s) indices
            mi=$(( ${#word} / 2 - 1 ))
        else
            mi=$(( ${#word} / 2 ))
        fi

        if (( ${#word} % 2 == 0 )); then # assign the middle letters
            ml="${word:$mi:2}"
        else
            ml="${word:$mi:1}"
        fi

        # modified Word
        if [ $num == 0 ];then
            mw=$word
        else
```

```
                mw="${word:0:$mi}$ml$(printf "%0.s$ml" $(seq 1 $(($num-1))))${word:$mi+${#ml}}"
            fi

            echo ""
            echo "Modified Word: $mw" # output

            echo "$word:$num:$mw" >> Alpha.txt
            break

          fi
        done
          ;;
    "log")
        echo ""
        if [ ! -s Alpha.txt ]; then
          echo "The log file is empty, run some calculations"
        else
          echo "Displaying the contents of the log file"
          echo ""
          cat Alpha.txt
        fi
        echo ""
        ;;
    "res")
        rm Alpha.txt
        touch Alpha.txt;;
    "9")
        echo ""
        echo "Thank you for using the Alpha Power Module, we hope you had a good experience!"
        echo ""
        echo "**********************************************************************************************************************************************************"
        echo ""
        break
        ;;
    *)
        echo "Invalid Input, Try Again!"
        ;;
  esac
done
```

```
215         ;;
216     "ep") # Expo Power case
217     while true # Inner While Controls the local menu
218     do
219       echo ""
220       echo "*****************************************************************************************************************************************************"
221       echo ""
222       echo -e "\e[5mWelcome to the \e[1mExpo Power Module!\e[0m"
223       echo ""
224       echo -e "\e[3mChoice\e[0m - Operation"
225       echo ""
226       echo -e "\e[3mcalc\e[0m - Use the Calculator"
227       echo -e "\e[3mlog\e[0m - View the Log file"
228       echo -e "\e[3mres\e[0m - Reset the Log file"
229       echo -e "\e[3m9\e[0m - Exit the module"
230       echo ""
231       read -p "Which operation would you like to do? " ch
232       case $(tr '[:upper:]' '[:lower:]' <<< "$ch") in # local case - esac construct
233         "calc")
234             i='y'
235             sno=1
236             while true
237             do
238               echo ""
239               read -p "Enter the base number: " base
240               if [ -z $base ]; then # empty condition
241                     echo "The base number is empty, try again!"
242                     continue
243               fi
244               if [[ $base =~ [0-9]+.[0-9]+ ]]; then # integer condition
245                 echo ""
246                 echo "The input should be an integer!"
247                 read -p "Do you want to try again (y/n)? " i
248                 if [ $(tr '[:upper:]' '[:lower:]' <<< "$i") == 'y' ]; then
249                   continue
250                 else
251                   break
252                 fi
253               fi
```

```
254             if ! (( $base >= 2 && $base <= 6 )); then # length range condition
255               echo ""
256               echo "The base number should be only between 2 and 6!"
257               read -p "Do you want to try again (y/n)? " i
258               if [ $(tr '[:upper:]' '[:lower:]' <<< "$i") == 'y' ]; then
259                 continue
260               else
261                 break
262               fi
263             elif [[ ! $base =~ [0-9]+ ]]; then # numerical condition
264               echo ""
265               echo "The base number should contain only numbers!"
266               read -p "Do you want to try again (y/n)? " i
267               if [ $(tr '[:upper:]' '[:lower:]' <<< "$i") == 'y' ]; then
268                 continue
269               else
270                 break
271               fi
272             else
273               read -p "Enter the exponent: " expo
274               if [ -z $expo ]; then # empty condition
275                     echo "The exponent is empty, try again!"
276                     continue
277               fi
278               if [[ $expo =~ [0-9]+.[0-9]+ ]]; then # integer condition
279                 echo ""
280                 echo "The input should be an integer!"
281                 read -p "Do you want to try again (y/n)? " i
282                 if [ $(tr '[:upper:]' '[:lower:]' <<< "$i") == 'y' ]; then
283                   continue
284                 else
285                   break
286                 fi
287               fi
288               if [[ ! $expo =~ [0-9]+ ]]; then # numerical condition
289                 echo ""
290                 echo "The exponent should contain only numbers!"
291                 read -p "Do you want to try again (y/n)? " i
292                 if [ $(tr '[:upper:]' '[:lower:]' <<< "$i") == 'y' ]; then
```

```
                continue
              else
                break
              fi
            fi
          fi

            read -p "Whats your guess for $base to the power of $expo ?" guess
            echo ""

            res=$(echo "scale=10; $base^$expo" | bc) # calculate the actual value

            if (( $(echo "$res == $guess" | bc -l) )); then # compare the result
              echo "Congratulations, you are a power player!"
            else
              echo "Game Over, you lose, low on power, time to recharge!"
            fi

            echo "$sno:$base:$expo:$guess:$res" >> LiPowerFile.txt

            ((sno++))
            break
        done
          ;;
    "log")
        echo ""
        if [ ! -s LiPowerFile.txt ]; then
          echo "The log file is empty, run some calculations"
        else
          echo "Displaying the contents of the log file"
          echo ""
          cat LiPowerFile.txt
        fi
        echo ""
        ;;
    "res")
        rm LiPowerFile.txt
        touch LiPowerFile.txt;;
    "9")
```

```
            echo ""
            echo "Thank you for using the Expo Power Module, we hope you had a good experience!"
            echo ""
            echo "**********************************************************************************************************************************************************"
            echo ""
            break
            ;;
        *)
            echo "Invalid Input, Try Again!"
            ;;
      esac
    done
    ;;
    "9")
        echo ""
        echo "Thank you for using the Power Calculator, we hope you had a good experience!"
        echo ""
        break
        ;;
    *)
        echo "Invalid Input, Try Again!"
        echo ""
        echo "**********************************************************************************************************************************************************"
        echo ""
        ;;
  esac
done

# Post Processing
rm Alpha.txt
rm LiPowerFile.txt
```

3 Conclusion

The code was run on multiple test cases, and it met all the required objectives.