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 9th 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 ************************************************************"
echo ""

echo -e "\e[5mWelcome to the \e[1mPower Calculator!\e[0m"

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 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 \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 "***********************************************************************************************************************************************************"
echo ""

# Actual Code

while true # Outer While controls the main menu
do
  echo -e "\e[5mWelcome to the \e[1mPower Calculator!\e[0m"
  echo ""
  echo -e "\e[1m\e[4:3mMain Menu\e[0m"
  echo ""
  echo -e "\e[3mChoice\e[0m - Operation"
  echo ""
  echo -e "\e[3mman\e[0m - Display Usage Notes"
  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”
        echo −e ”\e[3mlog\e[0m − View the Log file”
        echo −e ”\e[3mres\e[0m − Reset the Log file”
        echo −e ”\e[3m9\e[0m − Exit the module”
        echo ””
        read −p ”Which operation would you like to do? ” ch
        case $(tr '[:upper:]' '[:lower:]' <<< ”$ch”) in # local case − esac construct
          ”calc”)
            i='y'
            while true
            do
              echo ””
              read −p ”Enter the word to be operated on (7−9 characters only): ” word
              len=${#word}
              if [ −z $word ]; then # empty condition
                      echo ”Word is empty, try again!”
                      continue
              fi
              if ! (( $len >= 7 && $len <= 9 )); then # length range condition
                echo ””
                echo ”The word should be only 7−9 characters long!”
                read −p ”Do you want to try again (y/n)? ” i
                if [ $(tr '[:upper:]' '[:lower:]' <<< ”$i”) == 'y' ]; then
                  continue
                else
                  break
                fi
              elif [[ $word =~ [^[:alpha:]] ]]; then # alphabetical condition
                echo ””
                echo ”The word should contain only alphabets!”
                read −p ”Do you want to try again (y/n)? ” i
                if [ $(tr '[:upper:]' '[:lower:]' <<< ”$i”) == 'y' ]; then
                  continue
                else
                  break
                fi
              else
                read −p ”Enter the number of times the middle letter is to be displayed: ” num
                if [ −z $num ]; then # empty condition
                      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
```

```
        ;;
    "ep") # Expo Power case
    while true # Inner While Controls the local menu
    do
      echo ""
      echo "******************************************************************************************************************************************************************"
      echo ""
      echo -e "\e[5mWelcome to the \e[1mExpo Power Module!\e[0m"
      echo ""
      echo -e "\e[3mChoice\e[0m - Operation"
      echo ""
      echo -e "\e[3mcalc\e[0m - Use the Calculator"
      echo -e "\e[3mlog\e[0m - View the Log file"
      echo -e "\e[3mres\e[0m - Reset the Log file"
      echo -e "\e[3m9\e[0m - Exit the module"
      echo ""
      read -p "Which operation would you like to do? " ch
      case $(tr '[:upper:]' '[:lower:]' <<< "$ch") in # local case - esac construct
        "calc")
            i='y'
            sno=1
            while true
            do
              echo ""
              read -p "Enter the base number: " base
              if [ -z $base ]; then # empty condition
                    echo "The base number is empty, try again!"
                    continue
              fi
              if [[ $base =~ [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 ! (( $base >= 2 && $base <= 6 )); then # length range condition
  echo ""
  echo "The base number should be only between 2 and 6!"
  read -p "Do you want to try again (y/n)? " i
  if [ $(tr '[:upper:]' '[:lower:]' <<< "$i") == 'y' ]; then
    continue
  else
    break
  fi
elif [[ ! $base =~ [0-9]+ ]]; then # numerical condition
  echo ""
  echo "The base number 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
else
  read -p "Enter the exponent: " expo
  if [ -z $expo ]; then # empty condition
        echo "The exponent is empty, try again!"
        continue
  fi
  if [[ $expo =~ [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 [[ ! $expo =~ [0-9]+ ]]; then # numerical condition
    echo ""
    echo "The exponent 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
          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.