



REPORT

Group 2

Kanak Agarwal^{1,†}, Pranav Karkera^{1,‡}, Chirag Kumar Jain^{1,§} Shreyas V S^{1,||}, Tushar Raj ^{2,#}

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

 $^{^{\}dagger}$ Registration Number - 210933058

[‡] Registration Number - 210933046

 $[\]S$ Registration Number - 210933062

 $[\]parallel$ 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

```
        3
        # Assignment/FISAC

        4
        # Academic Year 2023-24

        5
        # Bath of 2021-25

        6
        # Group 2

        7
        # Code Developed on April 8th 2024

        8
        **

        10
        touch Alpha.txt

        11
        touch LiPowerFile.txt

        12
        **

        13
        # Header Write

        4
        **

        4
        **

        5
        **

        6
        **

        12
        **

        13
        **

        14
        **

        15
        **

        16
        **

        17
        **

        18
        **

        19
        **

        20
        **

        21
        **

        22
        **

        23
        **

        24
        **

        25
        **

        26
        **

        27
        **

        28
        **

        29
        **

        20
```

```
echo ""
21
   echo —e "\e[5mWelcome to the \e[1mPower Calculator!\e[0m"
23
   echo ""
24
25
   echo — e "\e[1m\e[4:3mUsage Notes\e[0m"]
   echo —e "The \e[3mPower Calculator\e[0m has the following submodules:"
   echo -e "\e[4mAlpha Power:\e[0m"
   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)."
34
   echo ""
35
36
   echo –e "\e[4mExpo Power:\e[0m"
   echo — "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)."]
41
   echo ""
42
   echo ""
45
   # Actual Code
   while true # Outer While controls the main menu
50
     echo –e "\e[5mWelcome to the \e[1mPower Calculator!\e[0m"
51
     echo —e "\e[1m\e[4:3mMain Menu\e[0m"
53
     echo ""
     echo —e "\e[3mChoice\e[0m — Operation"
     echo ""
56
     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"
60
    echo ""
    read -p "Which operation would you like to do?" c
62
    case $(tr '[:upper:]' '[:lower:]' <<< "$c") in # global case - esac construct
63
     "man")
64
     echo "
     66
     echo -e "\e[1m\e[4:3mUsage Notes\e[0m"
67
     echo ""
     echo —e "The \e[3mPower Calculator\e[0m has the following submodules:"
     echo "
70
     echo -e "\e[4mAlpha Power:\e[0m"
71
     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.
73
     A detailed log of each operation is stored in the file \e[3mAlpha\e[0m (.txt file)."
74
75
     echo ""
76
77
     echo —e "\e[4mExpo Power:\e[0m"
78
     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)."]
81
     echo ""
84
     "ap") # Alpha Power Case
        while true # Inner While Controls the local menu
        do
         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 "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
          _{
m else}
            break
        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
          \mathbf{fi}
        else
          read -p "Enter the number of times the middle letter is to be displayed:" num
          if [-z \text{ } num ]; then \# empty condition
                  echo "Number is empty, try again!"
```

100

101

102

103

105

106

107

109

110

111

112

113

114

116

117

118

119

120

121

123

124

126

127

128

130

131

133

134

135

136

```
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
if [[ \text{$num = \tilde{(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))
  mi=$(( ${#word} / 2 ))
fi
if ((\$\{\#word\}\ \%\ 2==0)); then \# assign the middle letters
  ml="${word:$mi:2}"
else
  ml="${word:$mi:1}"
_{\rm fi}
# modified Word
if [ \text{$num == 0 }];then
  mw=$word
```

137

138

139

140

141

142

144

145

146

148

149

150

151

152

153

155

156

157

158 159

160

161 162

163

164 165

166

167

169

170

173

174

 $_{
m else}$

```
mw = "\$\{word:0:\$mi\}\$ml\$(printf "\%0.s\$ml" \$(seq 1 \$((\$num-1))))\$\{word:\$mi+\$\{\#ml\}\}"\}
                    \mathbf{fi}
177
178
                    echo ""
179
                    echo "Modified Word: $mw" # output
180
181
                    echo "$word:$num:$mw" >> Alpha.txt
                    break
184
                   fi
                 done
187
             "\log")
                 echo ""
                 if [! -s Alpha.txt]; then
190
                   echo "The log file is empty, run some calculations"
191
                 else
192
                   echo "Displaying the contents of the log file"
                   echo ""
194
                   cat Alpha.txt
195
                 echo ""
                 ;;
198
              "res")
199
                 rm Alpha.txt
                 touch Alpha.txt;;
201
              "9")
202
                 echo ""
                 echo "Thank you for using the Alpha Power Module, we hope you had a good experience!"
204
                 echo ""
205
                 206
                 echo ""
                 break
208
                 ;;
209
                 echo "Invalid Input, Try Again!"
                 ;;
212
            esac
          done
```

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

```
if! (( base \ge 2 \&\& base \le 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
  _{
m 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
  \mathbf{fi}
 if [[\$\exp o = [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 [[! \$ \exp o = [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
```

254

255

256

257

258

259

261

262

263

265

266

267

268

269

270

272

273

274

275

276

277

279

280

281

282

283

284

286

287

289

290

291

292

```
continue
                        _{
m else}
294
                          break
295
                        \mathbf{fi}
                      fi
297
                    fi
298
                      read -p "Whats your guess for $base to the power of $expo?" guess
300
                      echo ""
301
302
                      res=$(echo "scale=10; $base^$expo" | bc) # calculate the actual value
303
304
                      if (( $(echo "$res == $guess" | bc -l) )); then # compare the result
305
                        echo "Congratulations, you are a power player!"
                      else
307
                        echo "Game Over, you lose, low on power, time to recharge!"
                      fi
309
310
                      echo "$sno:$base:$expo:$guess:$res" >> LiPowerFile.txt
311
312
                      ((sno++))
313
                      break
314
                 done
315
                   ;;
316
             "log")
317
                 echo ""
318
                 if [! -s LiPowerFile.txt]; then
319
                   echo "The log file is empty, run some calculations"
320
                 else
321
                   echo "Displaying the contents of the log file"
322
                   echo ""
323
                   cat LiPowerFile.txt
                 fi
325
                 echo ""
326
                 ;;
327
             "res")
328
                 rm LiPowerFile.txt
329
                 touch LiPowerFile.txt;;
330
             "9")
331
```

```
echo ""
          echo "Thank you for using the Expo Power Module, we hope you had a good experience!"
333
          echo ""
334
          335
          echo ""
336
          break
337
          ;;
339
          echo "Invalid Input, Try Again!"
340
          ;;
341
      esac
^{342}
     done
343
344
     "9")
       echo ""
346
       echo "Thank you for using the Power Calculator, we hope you had a good experience!"
347
       echo ""
348
       break
349
350
351
       echo "Invalid Input, Try Again!"
352
       echo ""
353
       354
       echo ""
355
    esac
357
   done
358
   # Post Processing
   rm Alpha.txt
361
   \rm rm\ LiPowerFile.txt
```

3 Conclusion

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