



---

# REPORT

## Introduction to Linux and Shell Scripting Optional Elective 2023-2024\*

### Group 2

Kanak Agarwal<sup>1,†</sup>, Pranav Karkera<sup>1,‡</sup>, Chirag Kumar Jain<sup>1,§</sup>  
Shreyas V S<sup>1,||</sup>, Tushar Raj<sup>2,#</sup>

---

<sup>1</sup> Department of Aeronautical and Automobile Engineering, Manipal Institute of Technology, Manipal, Karnataka, India - 576104

<sup>2</sup> Department of Electronics and Instrumentation Engineering, Manipal Institute of Technology, Manipal, Karnataka, India - 576104

\* Report Submitted on the 9<sup>th</sup> of April, 2024

<sup>†</sup> Registration Number - 210933058

<sup>‡</sup> Registration Number - 210933046

<sup>§</sup> Registration Number - 210933062

<sup>||</sup> Registration Number - 210933028

<sup>#</sup> 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

```
1  #!/bin/bash
2  # Optional Elective – Introduction to Linux and Shell Scripting
3  # Assignment/FISAC
4  # Academic Year 2023–24
5  # Batch of 2021–25
6  # Group 2
7  # Code Developed on April 8th 2024
8
9  # Pre Processing
10 touch Alpha.txt
11 touch LiPowerFile.txt
12
13 # Header Write
14 echo "Word:Number:Modified Word" >> Alpha.txt
15 echo "S_No:Base Number:Exponent:Guess:Answer" >> LiPowerFile.txt
16
17 # Banner
18 echo ""
19 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"

```

```

59  echo -e "\e[3mEP\e[0m - Launch Expo Power"
60  echo -e "\e[3m9\e[0m - Exit the Calculator"
61  echo ""
62  read -p "Which operation would you like to do? " c
63  case $(tr '[:upper:]' '[:lower:]' <<< "$c") in # global case - esac construct
64      "man")
65      echo ""
66      echo "*****"
67      echo -e "\e[1m\e[4:3mUsage Notes\e[0m"
68      echo ""
69      echo -e "The \e[3mPower Calculator\e[0m has the following submodules:"
70      echo ""
71      echo -e "\e[4mAlpha Power:\e[0m"
72      echo ""
73      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.
74      A detailed log of each operation is stored in the file \e[3mAlpha\e[0m (.txt file)."
75
76      echo ""
77
78      echo -e "\e[4mExpo Power:\e[0m"
79      echo ""
80      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).
81      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)."
82      echo "*****"
83
84      echo ""
85      ;;
86
87  "ap") # Alpha Power Case
88      while true # Inner While Controls the local menu
89      do
90          echo ""
91          echo "*****"
92          echo ""
93          echo -e "\e[5mWelcome to the \e[1mAlpha Power Module!\e[0m"
94          echo ""
95          echo -e "\e[3mChoice\e[0m - Operation"
96          echo ""
97          echo -e "\e[3mcalc\e[0m - Use the Calculator"

```



```

137         continue
138     fi
139     if [[ ! $num =~ [0-9]+ ]]; then # numerical condition
140         echo ""
141         echo "The input should contain only numbers!"
142         read -p "Do you want to try again (y/n)? " i
143         if [ $(tr '[:upper:]' '[:lower:]' <<< "$i") == `y` ]; then
144             continue
145         else
146             break
147         fi
148     fi
149     if [[ $num =~ [0-9]+.[0-9]+ ]]; then # integer condition
150         echo ""
151         echo "The input should be an integer!"
152         read -p "Do you want to try again (y/n)? " i
153         if [ $(tr '[:upper:]' '[:lower:]' <<< "$i") == `y` ]; then
154             continue
155         else
156             break
157         fi
158     fi
159
160     if (( ${#word} % 2 == 0 )); then # Calculate the middle letter(s) indices
161         mi=$(( ${#word} / 2 - 1 ))
162     else
163         mi=$(( ${#word} / 2 ))
164     fi
165
166     if (( ${#word} % 2 == 0 )); then # assign the middle letters
167         ml="${word:$mi:2}"
168     else
169         ml="${word:$mi:1}"
170     fi
171
172     # modified Word
173     if [ $num == 0 ];then
174         mw=$word
175     else

```

```
176         mw="{word:0:$mi}$ml$(printf "%0.s$ml" $(seq 1 $(( $num-1))))${word:$mi+${#ml}}}"
177     fi
178
179     echo ""
180     echo "Modified Word: $mw" # output
181
182     echo "$word:$num:$mw" >> Alpha.txt
183     break
184
185     fi
186 done
187 ;;
188 "log")
189     echo ""
190     if [ ! -s Alpha.txt ]; then
191         echo "The log file is empty, run some calculations"
192     else
193         echo "Displaying the contents of the log file"
194         echo ""
195         cat Alpha.txt
196     fi
197     echo ""
198     ;;
199 "res")
200     rm Alpha.txt
201     touch Alpha.txt;;
202 "g")
203     echo ""
204     echo "Thank you for using the Alpha Power Module, we hope you had a good experience!"
205     echo ""
206     echo "*****"
207     echo ""
208     break
209     ;;
210 *)
211     echo "Invalid Input, Try Again!"
212     ;;
213 esac
214 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

```

```

293         continue
294     else
295         break
296     fi
297 fi
298 fi
299
300 read -p "Whats your guess for $base to the power of $expo ?" guess
301 echo ""
302
303 res=$(echo "scale=10; $base^$expo" | bc) # calculate the actual value
304
305 if (( $(echo "$res == $guess" | bc -l) )); then # compare the result
306     echo "Congratulations, you are a power player!"
307 else
308     echo "Game Over, you lose, low on power, time to recharge!"
309 fi
310
311 echo "$sno:$base:$expo:$guess:$res" >> LiPowerFile.txt
312
313 ((sno++))
314 break
315 done
316 ;;
317 "log")
318     echo ""
319     if [ ! -s LiPowerFile.txt ]; then
320         echo "The log file is empty, run some calculations"
321     else
322         echo "Displaying the contents of the log file"
323         echo ""
324         cat LiPowerFile.txt
325     fi
326     echo ""
327 ;;
328 "res")
329     rm LiPowerFile.txt
330     touch LiPowerFile.txt;;
331 "g")

```

```
332     echo ""
333     echo "Thank you for using the Expo Power Module, we hope you had a good experience!"
334     echo ""
335     echo "*****"
336     echo ""
337     break
338 ;;
339 *)
340     echo "Invalid Input, Try Again!"
341 ;;
342 esac
343 done
344 ;;
345 "9")
346     echo ""
347     echo "Thank you for using the Power Calculator, we hope you had a good experience!"
348     echo ""
349     break
350 ;;
351 *)
352     echo "Invalid Input, Try Again!"
353     echo ""
354     echo "*****"
355     echo ""
356 ;;
357 esac
358 done
359
360 # Post Processing
361 rm Alpha.txt
362 rm LiPowerFile.txt
```

### 3 Conclusion

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