1. Write a program that takes a four-digit integer as a command line input. The script should give elegant message if the input is not an integer of right width or a float or a string. Convert the four-digit integer to a string based on the following rules for replacement:

Digit	Characters to replace
0	0
1	1
2	A or B or C
3	D or E or F
4	G or H or I
5	J or K or L
6	M or N or O
7	P or Q or R or S
8	T or U or V
9	W or X or Y or Z

Generate all the combinations of four-character words that are possible for the given input. [5 Mark]
Output required: The code, screenshots showing error response, list of words for two different inputs.
Application: One can remember a four-digit PIN using a word or acronym that could mean something for the user.

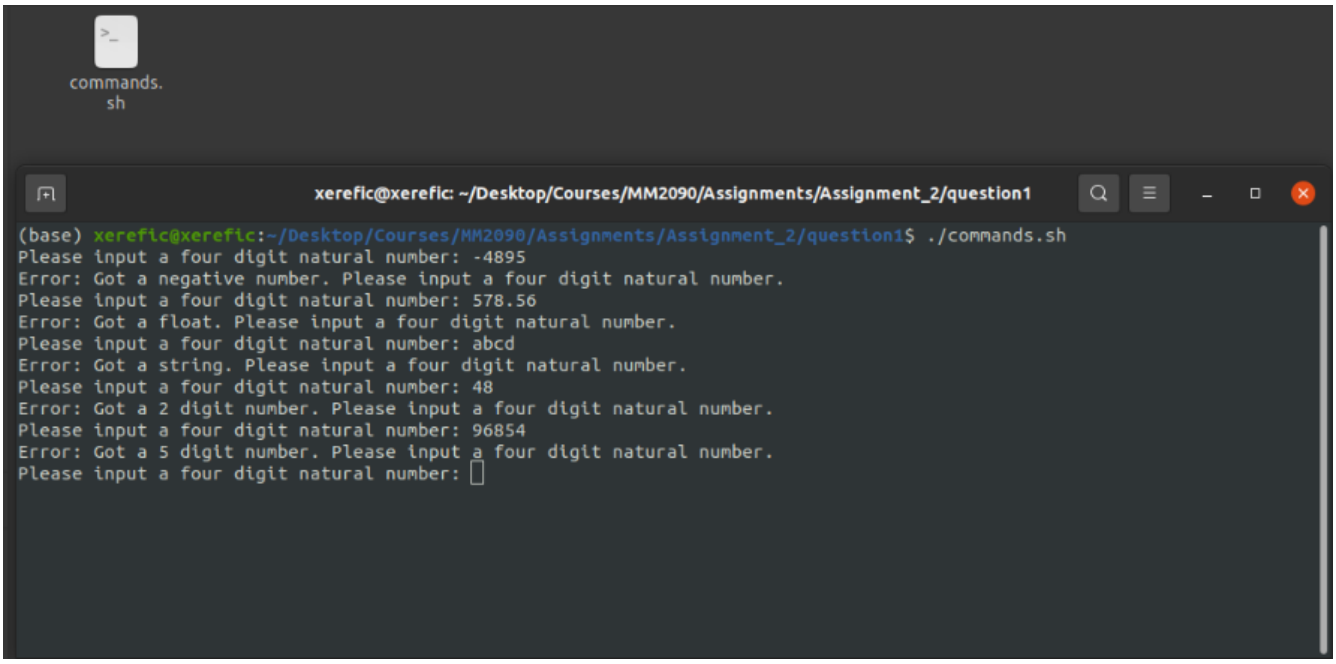
Link to the GitHub repository for this question: GitHub

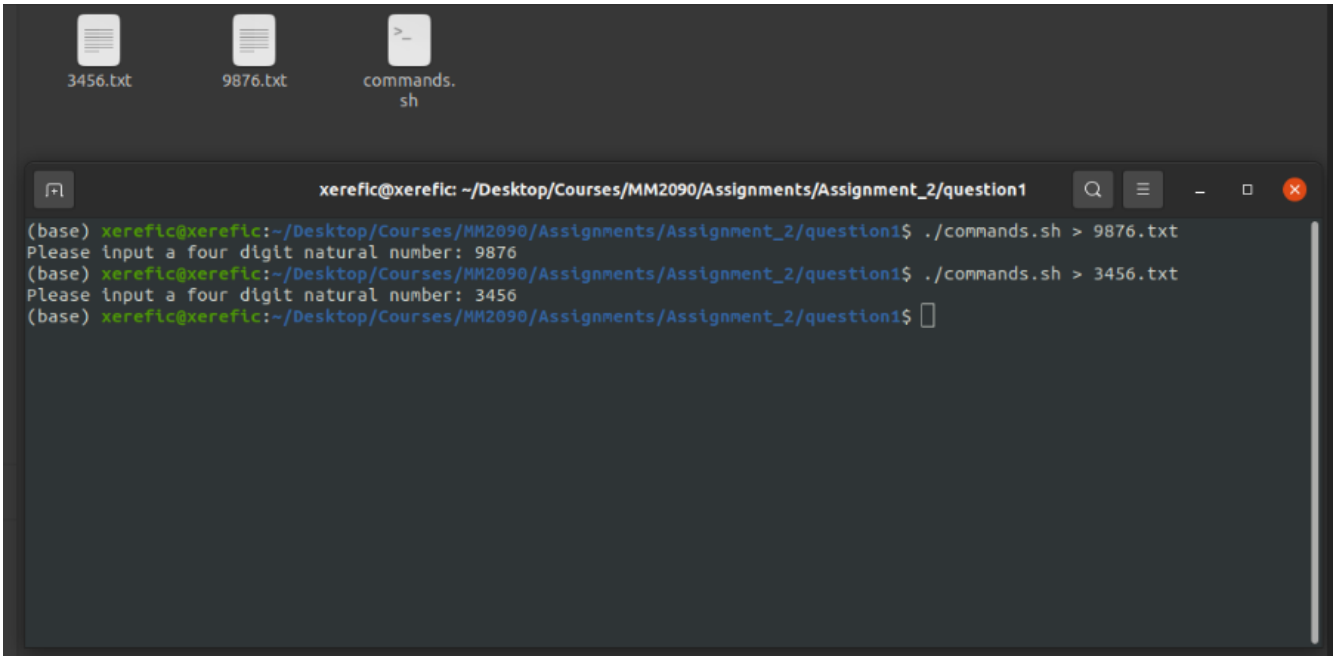
This bash script takes in a 4-digit natural number and encodes according to the given field.

```
#!/bin/bash

read -p "Please input a four digit natural number: " input


if [[ $input =~ ^-[0-9]+$ ]]; then
  echo "Error: Got a negative number. Please input a four digit natural number."
  ./$(basename $0) && exit

elif [[ $input =~ ^[0-9]+$ ]]; then
  if [[ ${#input} -ne 4 ]]; then
            echo "Error: Got a ${#input} digit number. Please input a four digit natural number."
            ./$(basename $0) && exit
  fi

elif [[ $input =~ ^[+-]?[0-9]+\.?[0-9]*$ ]]; then
  echo "Error: Got a float. Please input a four digit natural number."
  ./$(basename $0) && exit

else
  echo "Error: Got a string. Please input a four digit natural number."
  ./$(basename $0) && exit

fi


char0=("0")
char1=("1")
char2=("A" "B" "C")
char3=("D" "E" "F")
char4=("G" "H" "I")
char5=("J" "K" "L")
char6=("M" "N" "O")
char7=("P" "Q" "R" "S")
char8=("T" "U" "V")
char9=("W" "X" "Y" "Z")

typeset -n digit1=char${input:0:1}
typeset -n digit2=char${input:1:1}
typeset -n digit3=char${input:2:1}
```

```
41. typeset -n digit4=char${input:3:1}
42.
43. for c_1 in ${digit1[@]};
44. do
45.   for c_2 in ${digit2[@]};
46.   do
47.             for c_3 in ${digit3[@]};
48.             do
49.                       for c_4 in ${digit4[@]};
50.                       do
51.                                 echo $c_1$c_2$c_3$c_4
52.                       done;
53.             done;
54.   done;
55. done;
56.
```

TERMINAL:





OUTPUT (3456):

DGJM
DGJN
DGJO
DGKM
DGKN
DGKO
DGLM
DGLN
DGLO
DHJM
DHJN
DHJO
DHKM
DHKN
DHKO
DHLM
DHLN
DHLO
DIJM
DIJN
DIJO
DIKM
DIKN
DIKO
DILM
DILN
DILO
EGJM
EGJN
EGJO
EGKM
EGKN
EGKO
EGLM
EGLN
EGLO
EHJM
EHJN
EHJO
EHKM
EHKN

EHKO
EHLM
EHLN
EHLO
EIJM
EIJN
EIJO
EIKM
EIKN
EIKO
EILM
EILN
EILO
FGJM
FGJN
FGJO
FGKM
FGKN
FGKO
FGLM
FGLN
FGLO
FHJM
FHJN
FHJO
FHKM
FHKN
FHKO
FHLM
FHLN
FHLO
FIJM
FIJN
FIJO
FIKM
FIKN
FIKO
FILM
FILN
FILO

OUTPUT (9876):

WTPM
WTPN
WTPO
WTQM
WTQN
WTQO
WTRM
WTRN
WTRO
WTSM
WTSN
WTSO
WUPM
WUPN
WUPO
WUQM
WUQN
WUQO
WURM
WURN
WURO
WUSM
WUSN
WUSO
WVPM
WVPN
WVPO
WVQM
WVQN
WVQO
WVRM
WVRN
WVRO
WVSM
WVSN
WVSO
XTPM
XTPN
XTPO
XTQM
XTQN
XTQO
XTRM
XTRN
XTRO
XTSM
XTSN
XTSO

XUPM
XUPN
XUPO
XUQM
XUQN
XUQO
XURM
XURN
XURO
XUSM
XUSN
XUSO
XVPM
XVPN
XVPO
XVQM
XVQN
XVQO
XVRM
XVRN
XVRO
XVSM
XVSN
XVSO
YTPM
YTPN
YTPO
YTQM
YTQN
YTQO
YTRM
YTRN
YTRO
YTSM
YTSN
YTSO
YUPM
YUPN
YUPO
YUQM
YUQN
YUQO
YURM
YURN
YURO
YUSM
YUSN
YUSO

YVPM
YVPN
YVPO
YVQM
YVQN
YVQO
YVRM
YVRN
YVRO
YVSM
YVSN
YVSO
ZTPM
ZTPN
ZTPO
ZTQM
ZTQN
ZTQO
ZTRM
ZTRN
ZTRO
ZTSM
ZTSN
ZTSO
ZUPM
ZUPN
ZUPO
ZUQM
ZUQN
ZUQO
ZURM
ZURN
ZURO
ZUSM
ZUSN
ZUSO
ZVPM
ZVPN
ZVPO
ZVQM
ZVQN
ZVQO
ZVRM
ZVRN
ZVRO
ZVSM
ZVSN
ZVSO