

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.

```
1. #!/bin/bash
2.
3. read -p "Please input a four digit natural number: " input
4.
5.
6. if [[ $input =~ ^-[0-9]+$ ]]; then
7.     echo "Error: Got a negative number. Please input a four digit natural number."
8.     ./$(basename $0) && exit
9.
10. elif [[ $input =~ ^[0-9]+$ ]]; then
11.     if [[ ${#input} -ne 4 ]]; then
12.         echo "Error: Got a ${#input} digit number. Please input a four digit natural number."
13.         ./$(basename $0) && exit
14.     fi
15.
16. elif [[ $input =~ ^[+-]?[0-9]+\.[0-9]*$ ]]; then
17.     echo "Error: Got a float. Please input a four digit natural number."
18.     ./$(basename $0) && exit
19.
20. else
21.     echo "Error: Got a string. Please input a four digit natural number."
22.     ./$(basename $0) && exit
23.
24. fi
25.
26.
27. char0=("0")
28. char1=("1")
29. char2=("A" "B" "C")
30. char3=("D" "E" "F")
31. char4=("G" "H" "I")
32. char5=("J" "K" "L")
33. char6=("M" "N" "O")
34. char7=("P" "Q" "R" "S")
35. char8=("T" "U" "V")
36. char9=("W" "X" "Y" "Z")
37.
38. typeset -n digit1=char${input:0:1}
39. typeset -n digit2=char${input:1:1}
40. typeset -n digit3=char${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:

```

(base) xerefic@xerefic:~/Desktop/Courses/MM2090/Assignments/Assignment_2/question1$ ./commands.sh
Please input a four digit natural number: -4895
Error: Got a negative number. Please input a four digit natural number.
Please input a four digit natural number: 578.56
Error: Got a float. Please input a four digit natural number.
Please input a four digit natural number: abcd
Error: Got a string. Please input a four digit natural number.
Please input a four digit natural number: 48
Error: Got a 2 digit number. Please input a four digit natural number.
Please input a four digit natural number: 96854
Error: Got a 5 digit number. Please input a four digit natural number.
Please input a four digit natural number: 

```

```

(base) xerefic@xerefic:~/Desktop/Courses/MM2090/Assignments/Assignment_2/question1$ ./commands.sh > 9876.txt
Please input a four digit natural number: 9876
(base) xerefic@xerefic:~/Desktop/Courses/MM2090/Assignments/Assignment_2/question1$ ./commands.sh > 3456.txt
Please input a four digit natural number: 3456
(base) xerefic@xerefic:~/Desktop/Courses/MM2090/Assignments/Assignment_2/question1$ 

```

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

2. Create a make file that has the following behaviour when invoked as given below.

make	Output the usage pattern as help
make list	Recursively list all files in the current directory modified in the last n days
make backup	Copy all the files listed as above to a temporary directory and create a tar file for it. Name of the tar file shall be like backup-31May2021.tar if the command was invoked on 31 st May, 2021.

The value of n for the number of days should be configurable using a shell variable \$MODPERIOD. Default can be taken as 5 days. Except to copy the Makefile to a directory, the user is not expected to give any further input by hand. [5 Marks]

Output required: The code, screenshots showing its behaviour.

Application: One can place scripts in the cron directory to run them automatically at certain times. One can have a script there to back up the files that are being currently worked on to avoid accidental deletion.

Link to the GitHub repository for this question: [GitHub](#)

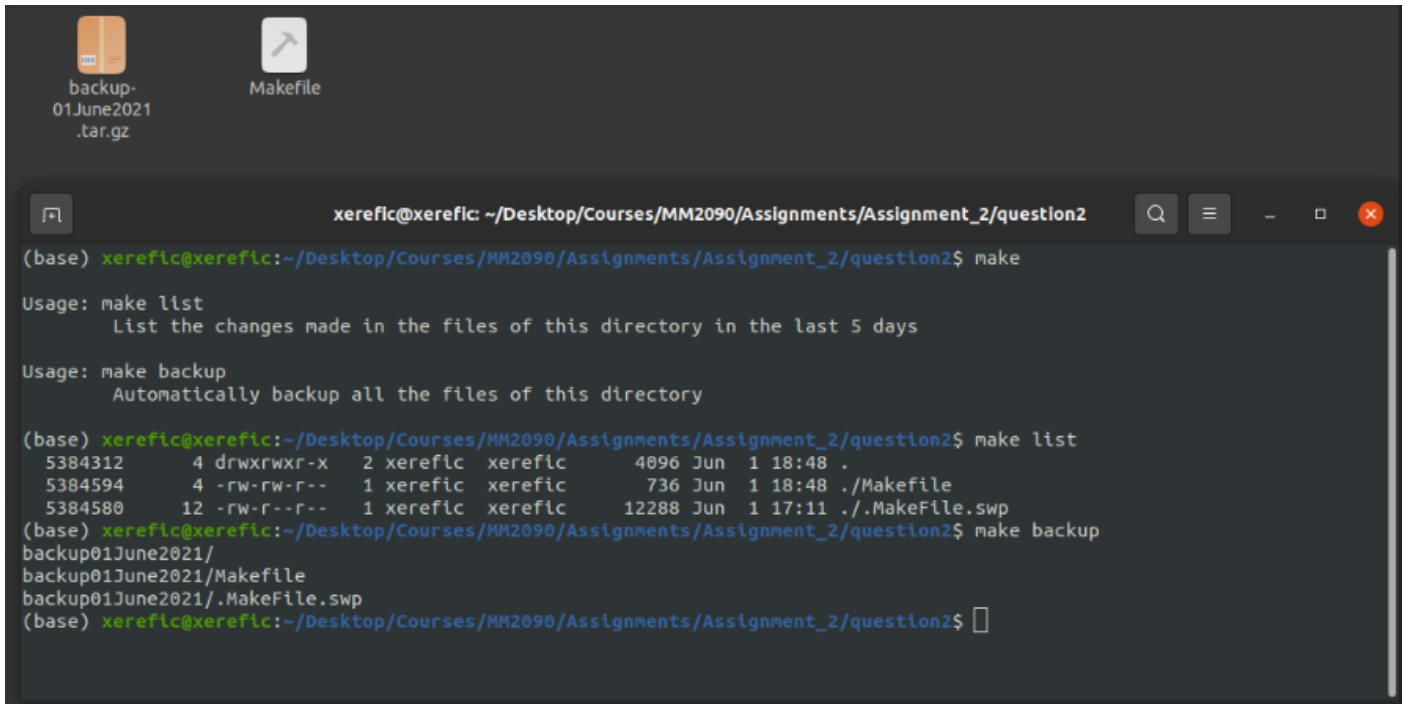
The make file takes automatically backs up the files edited in the last n days (5 days by default) and compresses it.

```

1. ifndef MODPERIOD
2.     MODPERIOD=5
3. endif
4.
5. DATE=`date | cut -d " " -f 2,3,4 | sed 's/ //g'`
6.
7. default:
8.     @echo
9.     @echo "Usage: make list"
10.    @echo "\tList the changes made in the files of this directory in the last $(MODPERIOD) days"
11.    @echo
12.    @echo "Usage: make backup"
13.    @echo "\tAutomatically backup all the files of this directory"
14.    @echo
15.
16. list:
17.    @find $(pwd) -mtime -$(MODPERIOD) -ls
18.
19. backup:
20.    @rm -f backup-$(DATE).tar
21.    @rm -rf ~/backup/backup$(DATE)/
22.    @mkdir ~/backup/backup$(DATE)
23.    @chmod 777 ~/backup/backup$(DATE)/
24.
25.    @cp -r `find $(pwd) -mtime -$(MODPERIOD)` ~/backup/backup$(DATE)/
26.    @cp -r ~/backup/backup$(DATE)/ ./backup$(DATE)/
27.    @rm -r ~/backup/backup$(DATE)/
28.
29.    @tar cvfz backup-$(DATE).tar.gz backup$(DATE)/
30.
31.    @rm -r ./backup$(DATE)/
32.

```

TERMINAL:



```
(base) xereflc@xereflc:~/Desktop/Courses/MM2090/Assignments/Assignment_2/question2$ make

Usage: make list
    List the changes made in the files of this directory in the last 5 days

Usage: make backup
    Automatically backup all the files of this directory

(base) xereflc@xereflc:~/Desktop/Courses/MM2090/Assignments/Assignment_2/question2$ make list
5384312      4 drwxrwxr-x   2 xereflc  xereflc    4096 Jun  1 18:48 .
5384594      4 -rw-rw-r--   1 xereflc  xereflc     736 Jun  1 18:48 ./Makefile
5384580     12 -rw-r--r--   1 xereflc  xereflc   12288 Jun  1 17:11 ../MakeFile.swp
(base) xereflc@xereflc:~/Desktop/Courses/MM2090/Assignments/Assignment_2/question2$ make backup
backup01June2021/
backup01June2021/Makefile
backup01June2021/MakeFile.swp
(base) xereflc@xereflc:~/Desktop/Courses/MM2090/Assignments/Assignment_2/question2$
```

3. Pick one “flavor” of Linux distribution (preferably unique in your group) and trace the timeline of its development. [3 Marks]

Output: Year wise release versions with names if applicable, hardware platforms supported, desktop environments available, kernel versions supported, one USP if applicable.

Application: One should be aware of specialized operating systems that come bundled with applications for a specific domain of usage. It helps get work done faster.

Linux¹

Linux is a family of [open-source Unix-like operating systems](#) based on the [Linux kernel](#). Distributions include the Linux kernel and supporting [system software](#) and [libraries](#), many of which are provided by the [GNU Project](#).

History²

Precursors

The [Unix](#) operating system was conceived and implemented in 1969, at [AT&T's Bell Labs](#), in the United States by [Ken Thompson](#), [Dennis Ritchie](#), [Douglas McIlroy](#), and [Joe Ossanna](#). First released in 1971, Unix was written entirely in [assembly language](#), as was common practice at the time. In 1973 in a key, pioneering approach, it was rewritten in the [C](#) programming language by [Dennis Ritchie](#). The availability of a [high-level language](#) implementation of Unix made its [porting](#) to different computer platforms easier.

The [GNU Project](#), started in 1983 by [Richard Stallman](#), had the goal of creating a "complete Unix-compatible software system" composed entirely of [free software](#). Work began in 1984.

[MINIX](#) was created by [Andrew S. Tanenbaum](#), a [computer science](#) professor, and released in 1987 as a minimal [Unix-like](#) operating system targeted at students and others who wanted to learn operating system principles.

Creation

In 1991, Torvalds became curious about operating systems. Frustrated by the licensing of MINIX, which at the time limited it to educational use only, he began to work on his own operating system kernel, which eventually became the [Linux kernel](#).

Distributions³ (Most Popular in 2021)

- MX Linux
- Manjaro
- Linux Mint
- Ubuntu
- Debian
- Elementary OS
- Solus
- Zorin OS
- Fedora
- Deepin

¹ [Source: Wikipedia](#)

² [Source: Wikipedia](#)

³ [Source](#)

Kali Linux⁴

Kali Linux is a [Debian](#)-derived [Linux distribution](#) designed for [digital forensics](#) and [penetration testing](#).^[3] It is maintained and funded by [Offensive Security](#).

Development⁵

It was developed by Mati Aharoni and Devon Kearns of Offensive Security through the rewrite of [BackTrack](#), their previous information security testing Linux distribution based on [Knoppix](#). Originally, it was designed with a focus on kernel auditing, from which it got its name **Kernel Auditing Linux**.

Kali Linux has around 600 pre-installed penetration-testing programs (tools), including:

[Armitage](#), [Nmap](#), [Wireshark](#), [metasploit](#), [John the Ripper](#), sqlmap, [Aircrack-ng](#), Burp suite and [OWASP ZAP web application security scanners](#), etc.

System Requirements⁶

- A minimum of 20GB hard disk space for installation depending on the version
- A minimum of 2GB RAM for i386 and AMD64 architectures.
- A bootable CD-DVD drive or a USB stick.

Supported Platforms⁷

Kali Linux is distributed in [32-bit](#) and [64-bit](#) images for use on hosts based on the [x86 instruction set](#) and as an image for the [ARM architecture](#) for use on the [Beagle Board](#) computer and Samsung's ARM [Chromebook](#).

Kali Linux is already available for Asus Chromebook Flip C100P, [BeagleBone Black](#), HP [Chromebook](#), CubieBoard 2, [CuBox](#), [CuBox-i](#), [Raspberry Pi](#), EfikaMX, Odroid U2, Odroid XU, Odroid XU3, [Samsung Chromebook](#), Utilite Pro, [Galaxy Note 10.1](#), and SS808.

With the arrival of [Kali NetHunter](#), Kali Linux is also officially available on Android devices such as the Nexus 5, Nexus 6, Nexus 7, Nexus 9, Nexus 10, OnePlus One, and some Samsung Galaxy models.

Kali Linux is available on [Windows 10](#), on top of [Windows Subsystem for Linux](#) (WSL). The official Kali distribution for Windows can be downloaded from the [Microsoft Store](#).

Desktop Environments⁸

- ✓ [XFCE](#)
- ✓ [KDE](#)
- ✓ [LXDE](#)
- ✓ [GNOME](#)
- ✓ [Cinnamon](#)
- ✓ [MATE](#)

⁴ [Source: Wikipedia](#)

⁵ [Source: Wikipedia](#)

⁶ [Source](#)

⁷ [Source: Wikipedia](#)

⁸ [Setup](#)

Version History⁹ and Kernels

Version Number	Release Date	Supported Kernel	Changes
Kali 1.0.0	13th March, 2013		Initial release, "moto".
Kali 2.0	11th August, 2015		Major release, "safi". Now a rolling distribution, major UI changes
Kali 2016.1	21st January, 2016	Kernel 4.3, GNOME 3.18.	The first Kali Rolling release
Kali 2017.1	25th April, 2017	Kernel 4.9, GNOME 3.22.	The first 2017 Kali Rolling release.
Kali 2018.1	6th February, 2018	Kernel 4.14.12, GNOME 3.26.2.	The first 2018 Kali Rolling release.
Kali 2019.1	18th February, 2019	Kernel 4.19.13, GNOME 3.30.2.	The first 2019 Kali Rolling release.
Kali 2020.1	28th January, 2020	Kernel 5.4.0, Xfce 4.14.2.	The first 2020 Kali Rolling release.
Kali 2021.1	24th February, 2021	Kernel 5.10.0, Xfce 4.16.1.	The first 2021 Kali Rolling release.
Kali 2021.2	1st June, 2021	Kernel 5.10.0, Xfce 4.16.2.	The second 2021 Kali Rolling release.

Unique Selling Proposition (USP)¹⁰

Kali Linux is specifically geared to meet the requirements of professional penetration testing and security auditing.

1. **Network services disabled by default:** Kali Linux contains systemd hooks that [disable network services](#) by default. These hooks allow us to install various services on Kali Linux, while ensuring that our distribution remains secure by default, no matter what packages are installed. Additional services such as Bluetooth are also blacklisted by default.
2. **Custom Linux kernel:** Kali Linux uses an upstream kernel, patched for wireless injection.
3. **A *minimal* and *trusted* set of repositories:** given the aims and goals of Kali Linux, maintaining the integrity of the system as a whole is absolutely key. With that goal in mind, the set of upstream software sources which Kali uses is [kept to an absolute minimum](#). Many new Kali users are tempted to add additional repositories to their **sources.list**, but doing so runs a *very serious risk* of breaking your Kali Linux installation.

⁹ [Source](#)

¹⁰ [Source](#)