

Q1: Read man ls and write an ls command that lists files in the following manner

Includes all files, including hidden files

Sizes are listed in human readable format (e.g. 454M instead of 454279954)

Files are ordered by recency

Output is colorized

Ans:

```
missing-semester — Faran Title — -bash — 80x24
[(base) 192:missing-semester mac$ ls -lathG
total 216
drwxr-xr-x 13 mac staff 416B Mar 11 18:01 .git
drwxr-xr-x 7 mac staff 224B Mar 11 17:50 Faran-Taimoor-Butt
-rw-r--r--@ 1 mac staff 8.0K Mar 11 17:48 .DS_Store
drwxr-xr-x@ 30 mac staff 960B Mar 11 17:48 .
drwxr-xr-x 4 mac staff 128B Mar 11 17:48 ..
drwxr-xr-x 5 mac staff 160B Mar 10 21:52 static
-rw-r--r-- 1 mac staff 24B Mar 10 21:52 robots.txt
-rw-r--r-- 1 mac staff 1.8K Mar 10 21:52 license.md
-rw-r--r-- 1 mac staff 58B Mar 10 21:52 lectures.html
-rw-r--r-- 1 mac staff 5.0K Mar 10 21:52 index.md
-rw-r--r-- 1 mac staff 7.2K Mar 10 21:52 favicon.ico
-rw-r--r-- 1 mac staff 3.8K Mar 10 21:52 favicon-32x32.png
-rw-r--r-- 1 mac staff 3.5K Mar 10 21:52 favicon-16x16.png
-rw-r--r-- 1 mac staff 207B Mar 10 21:52 docker-compose.yml
-rw-r--r-- 1 mac staff 7.0K Mar 10 21:52 apple-touch-icon.png
-rw-r--r-- 1 mac staff 6.5K Mar 10 21:52 about.md
drwxr-xr-x 6 mac staff 192B Mar 10 21:52 _layouts
drwxr-xr-x 7 mac staff 224B Mar 10 21:52 _includes
-rw-r--r-- 1 mac staff 452B Mar 10 21:52 _config.yml
drwxr-xr-x 16 mac staff 512B Mar 10 21:52 _2020
drwxr-xr-x 21 mac staff 672B Mar 10 21:52 _2019
-rw-r--r-- 1 mac staff 1.4K Mar 10 21:52 README.md
```

Q2: Write bash functions marco and polo that do the following. Whenever you execute marco the current working directory should be saved in some manner, then when you execute polo, no matter what directory you are in, polo should cd you back to the directory where you executed marco. For ease of debugging you can write the code in a file marco.sh and (re)load the definitions to your shell by executing source marco.sh.

```
HW3 — Faran Title — nano marco.sh — 80x24
UW PICO 5.09 File: marco.sh

macro() {
    export PresentDir=$(pwd)
}

polo() {
    cd "$PresentDir"
}

^G Get Help ^O WriteOut ^R Read File ^Y Prev Pg ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where is ^V Next Pg ^U UnCut Text ^T To Spell
```

```
HW3 — Faran Title — -bash — 80x24
(base) 192:HW3 mac$ ls
macro.sh
(base) 192:HW3 mac$ pwd
/Users/mac/Desktop/advance-python-mipt/missing-semester/Faran-Taimoor-Butt/HW3
(base) 192:HW3 mac$ source macro.sh
(base) 192:HW3 mac$ macro
(base) 192:HW3 mac$ cd ..
(base) 192:Faran-Taimoor-Butt mac$ cd ..
(base) 192:missing-semester mac$ ls
404.html          _2020              favicon-32x32.png
CNAME             _config.yml       favicon.ico
Dockerfile        _includes         index.md
Faran-Taimoor-Butt _layouts          lectures.html
Gemfile           about.md          license.md
Gemfile.lock      apple-touch-icon.png robots.txt
README.md         docker-compose.yml static
_2019             favicon-16x16.png
(base) 192:missing-semester mac$ polo
(base) 192:HW3 mac$ ls
macro.sh
(base) 192:HW3 mac$
```

Q3: Say you have a command that fails rarely. In order to debug it you need to capture its output but it can be time consuming to get a failure run. Write a bash script that runs the following script until it fails and captures its standard output and error streams to files and prints everything at the end. Bonus points if you can also report how many runs it took for the script to fail.

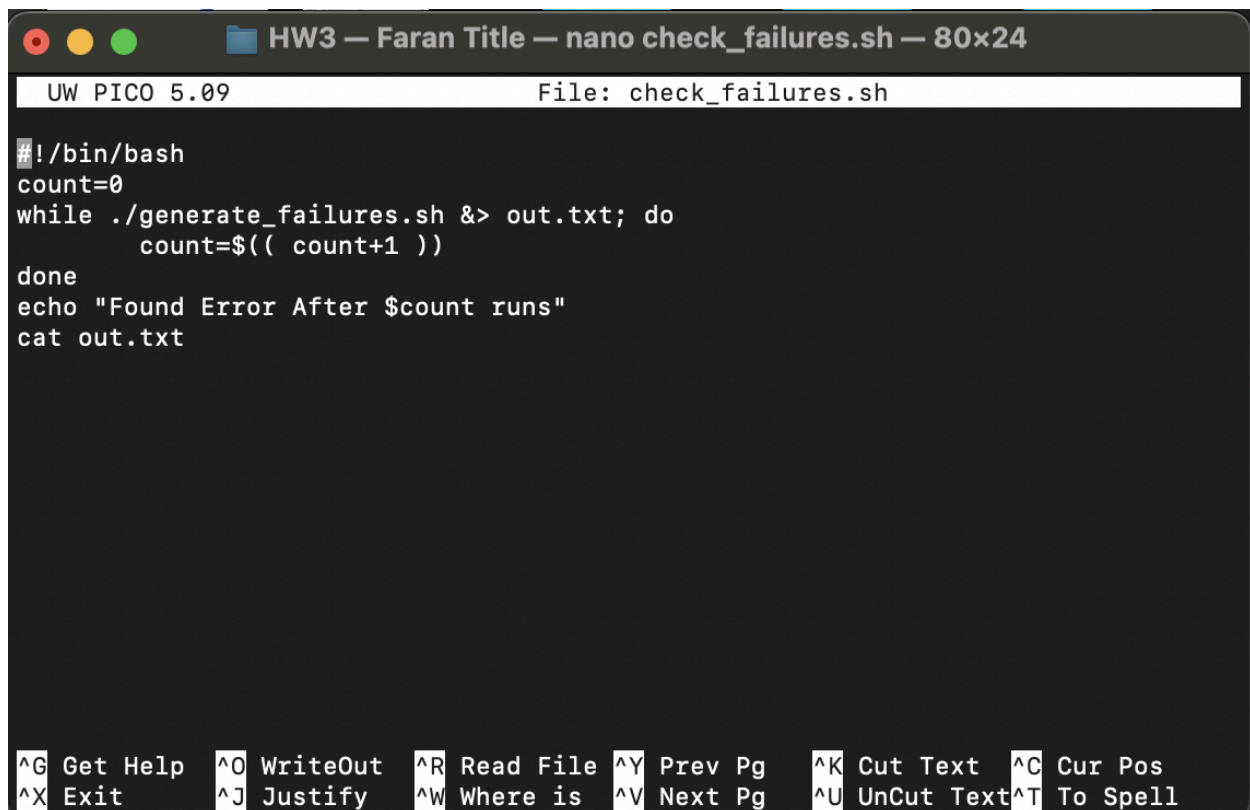
generate_failures.sh

```
HW3 — Faran Title — nano generate_failures.sh — 80x24
UW PICO 5.09      File: generate_failures.sh

#!/bin/bash
n=$((RANDOM % 100))
echo "$n"
if [[ $n -eq 42 ]]; then
    echo "Something went wrong"
    >&2 echo "The error was using magic numbers"
    exit 1
fi
echo "Every thing went right"

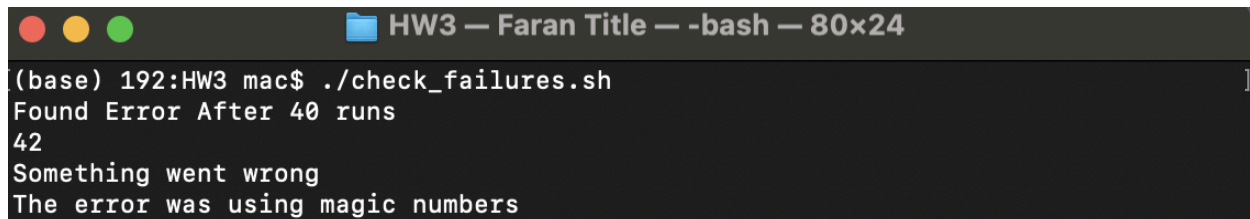
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Pg   ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify   ^W Where is   ^V Next Pg   ^U UnCut Text ^T To Spell
```

Check_failures.sh



```
#!/bin/bash
count=0
while ./generate_failures.sh &> out.txt; do
    count=$(( count+1 ))
done
echo "Found Error After $count runs"
cat out.txt
```

Check Failures by executing `check_failures.sh`



```
(base) 192:HW3 mac$ ./check_failures.sh
Found Error After 40 runs
42
Something went wrong
The error was using magic numbers
```

Q4:As we covered in the lecture find's `-exec` can be very powerful for performing operations over the files we are searching for. However, what if we want to do something with all the files, like creating a zip file? As you have seen so far commands will take input from both arguments and STDIN. When piping commands, we are connecting STDOUT to STDIN, but some commands like `tar` take inputs from arguments. To bridge this disconnect there's the `xargs` command which will execute a command using STDIN as arguments. For example `ls | xargs rm` will delete the files in the current directory.

Your task is to write a command that recursively finds all HTML files in the folder and makes a zip with them. Note that your command should work even if the files have spaces (hint: check `-d` flag for `xargs`). `{% comment %} find . -type f -name "*.html" | xargs -d '\n' tar -cvzf archive.tar.gz {% endcomment %}`

If you're on macOS, note that the default BSD find is different from the one included in GNU coreutils. You can use `-print0` on find and the `-0` flag on xargs. As a macOS user, you should be aware that command-line utilities shipped with macOS may differ from the GNU counterparts; you can install the GNU versions if you like by using brew.

Ans:

```
$ package_html.sh U x
Faran-Taimoor-Butt > HW3 > $ package_html.sh
1  #!/bin/bash
2
3  find . -type f -name '*.html' -print0 | xargs -0 tar -cvzf html_archive.tar.gz
4

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ... bash - HW3 + - [ ] [ ] ... ^ x
• (base) 192:HW3 mac$ ls
check_failures.sh      htmls                out.txt
generate_failures.sh   macro.sh             package_html.sh
• (base) 192:HW3 mac$ bash package_html.sh
a ./htmls/4.html
a ./htmls/3.html
a ./htmls/2.html
a ./htmls/1.html
○ (base) 192:HW3 mac$
```

HW3	●
htmls	●
<> 1.html	U
<> 2.html	U
<> 3.html	U
<> 4.html	U
\$ check_failures.sh	U
\$ generate_failures.sh	U
≡ html_archive.tar.gz	U
\$ macro.sh	U
≡ out.txt	U
\$ package_html.sh	U

Q5:(Advanced) Write a command or script to recursively find the most recently modified file in a directory. More generally, can you list all files by recency?

```
$ most_recent_file.sh U X
Faran-Taimoor-Butt > HW3 > $ most_recent_file.sh
1  #!/bin/bash
2
3  find . -type f -exec stat -f "%m %N" {} + | sort -n | tail -1
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
● (base) 192:HW3 mac$ bash most_recent_file.sh
1741778534 ./most_recent_file.sh
```

As most recent file is the script “**most_recent_file.sh**” that was written to find the most recent file

All most recent files:

```
$ all_most_recent_files.sh U X
Faran-Taimoor-Butt > HW3 > $ all_most_recent_files.sh
1  #!/bin/bash
2
3  find . -type f -exec stat -f "%m %N" {} + | sort -nr | cut -d ' ' -f2-
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  ...  bash - HW3
● (base) 192:HW3 mac$ bash all_most_recent_files.sh
./all_most_recent_files.sh
./most_recent_file.sh
./html_archeive.tar.gz
./package_html.sh
./htmls/4.html
./htmls/3.html
./htmls/2.html
./htmls/1.html
./out.txt
./check_failures.sh
./generate_failures.sh
./macro.sh
```