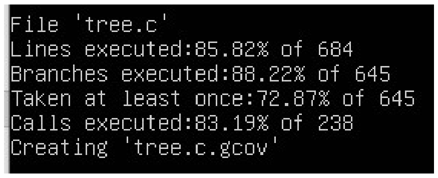
Question 1

C. Refer to `2018320250.sh`

D. Refer to screenshot below.



```
File 'tree.c'
Lines executed:85.82% of 684
Branches executed:88.22% of 645
Taken at least once:72.87% of 645
Calls executed:83.19% of 238
Creating 'tree.c.gcov'
```

E. All functions were covered by the script.

a. Refer to `2018320250.sh`

b. Ensure that `tree.c` and `2018320250.sh` are within in the same directory. Simply execute the command `./2018320250.sh` in the terminal (assumed to be `bash`) in the directory with the aforementioned files (ensure that the user has sufficient permissions to run the file; there will be further discussion on permissions and their impact on testing the shell script below). Running the same script twice will increase coverage when other files are not present (due to branches taken when files already exist), even though they were included in the submitted `zip` file.

Note that some common programs such as `gcc`, `mkdir`, `ln` etc. are assumed to be available to the user.

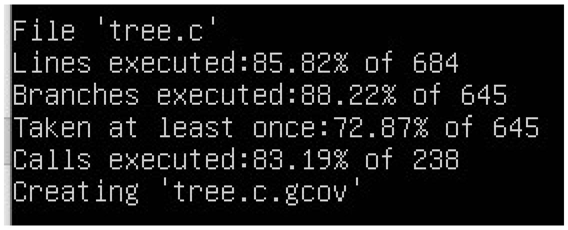
It is preferable to execute the bash script on a computer where the directory `/dev` has sufficiently many files of different file types to maximise coverage (i.e. the script was mostly tested and developed on a physical machine, in addition to a virtual machine running a full Ubuntu installation; minimalistic settings may reduce coverage).

The script also runs on the parent directory `..` for some of the test cases, hence executing the bash script in an appropriate directory may ease the reproduction of results.

While branch coverage was increased by testing the program under situations whose reproduction would require superuser privileges (i.e. by running `chmod` on certain files to test the program or by running the program on directories that may be inaccessible to the user etc.), the availability of such privilege was not assumed. Hence, there may be minor discrepancies in coverage when run on the machine used for grading.

However, minimal permissions such as those required to create files and directories were in fact assumed and are necessary to extend coverage to certain functions in the program (cf. the submitted bash script).

c. Screenshot of output below; based on tests run on the virtual machine; minor discrepancies may exist due to differences in set-up, permissions etc.



```
File 'tree.c'
Lines executed:85.82% of 684
Branches executed:88.22% of 645
Taken at least once:72.87% of 645
Calls executed:83.19% of 238
Creating 'tree.c.gcov'
```

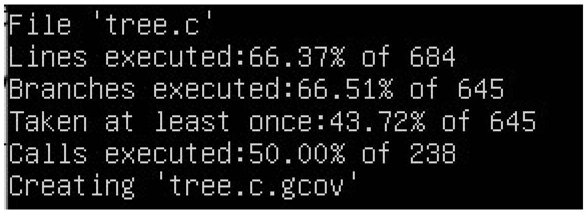
=> statements executed: **587** (out of 684)

d. Refer to screenshot above. Based on tests run on the virtual machine; minor discrepancies may exist due to differences in set-up, permissions etc.

=> branches executed: **569** (out of 645); branches taken at least once: **470** (out of 645) => 569 – 470 = **99** branches executed but not taken

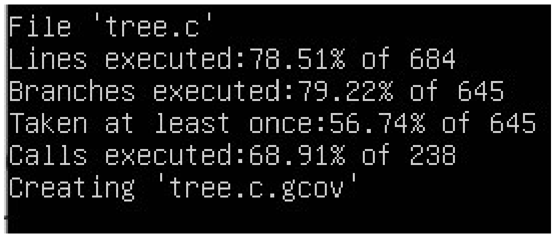
e. While a test case can be constructed in such a way that an arbitrary number of test cases is run, when the number of commands is restricted to one, the most effective test case was the test case where multiple options were used at the same time:

```
./tree -adlfiqNpugsDFtxASnC -H . -L 50 -R -o logs/log . ..
```

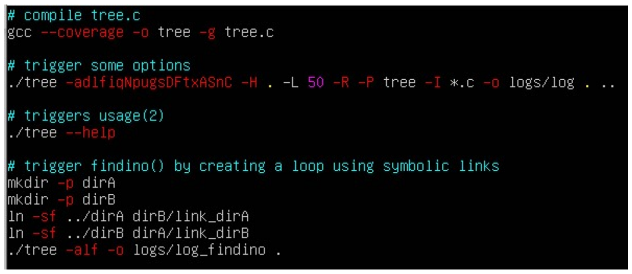


```
File 'tree.c'
Lines executed:66.37% of 684
Branches executed:66.51% of 645
Taken at least once:43.72% of 645
Calls executed:50.00% of 238
Creating 'tree.c.gcov'
```

f. Through some ablation testing (non-exhaustive), the three cases in `2018320250_t.sh` achieved the greatest branch coverage:



```
File 'tree.c'
Lines executed:78.51% of 684
Branches executed:79.22% of 645
Taken at least once:56.74% of 645
Calls executed:68.91% of 238
Creating 'tree.c.gcov'
```



```
# compile tree.c
gcc --coverage -o tree -g tree.c

# trigger some options
./tree -adlfiqNpugsDFtxASnC -H . -L 50 -R -P tree -I *.c -o logs/log . ..

# triggers usage(2)
./tree --help

# trigger findino() by creating a loop using symbolic links
mkdir -p dirA
mkdir -p dirB
ln -sf ../dirA dirB/link_dirA
ln -sf ../dirB dirA/link_dirB
./tree -alf -o logs/log_findino .
```

g. All functions were covered by the script.

h. All functions were covered by the script.