Introduction

This quiz is designed to be open-book and instructional. We strongly encourage you to use the questions here as learning exercises, instead of just answering the questions using your current knowledge.

Shebang options

The first line of a shell program has #!, which is called shebang. It specifies the interpreter to be used. Let us say that there are two executable files a.sh and b.sh, that are identical, except for the first lines shown below.

a. `#!/bin/bash`
b. `#!/usr/bin/env bash`

Which of the following statements are true?

a. Both the files behave the same way when executed.
b. For some people, depending on the environment variable settings, they can differ.
c. When invoked with sh (sh a.sh and sh b.sh), first will execute using sh whereas second one will invoke bash.
d. When invoked with sh (sh a.sh and sh b.sh), both the files will be executed with sh.

Possible answers are:

1. a and d
2. b and d
3. a and c
4. c and d
5. a, b, and d

Debugging shell scripts

Suppose we want to debug a shell script. We can do the following:

a. Execute it with -x option.
b. Execute it with -v option.
c. Surround the code with `set +x` and `set -x`.

Which of the following are true?

1. All of them work.
2. a alone works.
3. c works best, followed by a, and then b.
4. a works best, followed by b.
5. a and c work equally well.

Add function

Consider the following snippet of the code:

```
function add { if [ $# -ge 2 ]; then  echo `expr $1 + $2`;
  elif [ $# -ge 1 ]; then echo $1; else  echo 0; fi }
```

Consider the following functions:

a.
```
function add1 {
 case "$#" in
 0) echo 0;;
 1) echo $1;;
 *) echo `expr $1+$2`;;
 esac
}
```

b. ```bash function add2 { [[$# -ge 2]] && echo $(($1 + $2)) [[$# -eq 1]] && echo $1 [[$# -eq 0]] && echo 0 }

```
c. ```bash
function add3 { echo $(( ${1:-0} + ${2:-0} )); }
```

Which of the following statements are correct?

1. Only add2 and add1 are the same as add; not add3.
2. All add2 and add3 and add1 are all the same add.
3. Only add3 and add2 are the same as add; not add1.
4. Only add1 and add3 are the same as add not add2.
5. None of the functions are same as add.

getopts

If you are writing a program in bash, it is customary that you take options. Usually, they can be short options (-h, -v etc) or long options (–help, –verbose etc).

d. The invocation of this program “a.sh -vo f1” is the same as “a.sh -ov f1” which the same as “a.sh -v -o f1”.

1. All the statements are correct.
2. b and c are correct.
3. c and d are correct.
4. a and c are correct.
5. a and b are correct.

Filename normalization

You have a folder full of files. These file names have blanks and a mix of upper and lower cases. Your job is to write function that given a folder, renames all the files, replacing blanks with “-” and making all of them lowercase.

Consider the following segment:

```
function rename_files {
# Rename the files given as follows:
# one or more empty spaces go to a single _.
# And, all upper case
# go to lower case.
  for i in "$@"
  do
    mv $i $(echo $i | sed -e s/ /_/ | tr A-Z a-z)
  done
}
```

Consider all the suggestions to modify it.

a. Change `tr A-Z a-z` to `tr [A-Z] [a-z]`
b. Change `sed -e s/ /_/` to `sed -e 's/ \+/_/g'`.
c. Change `$i` to `"$i"` wherever it occurred.
d. Change `tr A-Z a-z` to `sed -e 's/./\L&/g'`

Which of the following statements is correct?

1. a is bad advice.
2. a and d are bad advices.
3. a and c and d are bad advices.
4. d is a bad advice.

5. a and b are bad advices.

Remote execution

Suppose you are working on some remote aws servers through a program you wrote. You need to execute a command on these 5 servers (call them aws1, aws2, aws3, asw4, aws5). After all of them are done, you should clean them up, which frees all the resources. Eventually, print a message and exit.

Here is a program that you have

```
for i in $(seq 5)
do
  remote_execute aws$i
done

for i in $(seq 5)
do
  remote_cleanup aws$i
done

echo "Done with AWS execution. Cleaned up after myself too."
```

The following statements have been made about this code:

a. The code is too sequential (and hence slow). If you do `remote_execute aws$i &` you will keep the program correct, but make it faster.
b. The code is fine as is. There is no problem with it.
c. You could trap any abnormal signals and do the remote cleanup, so that you don't pay for resources that you are not going to use.
d. You should replace `i` with `{i}` by surrounding with '{}'

Now, which of the statements is right?

1. b is correct.
2. c is correct.
3. a is correct.
4. a and c are correct.
5. a and c and d are correct.

trap

Read the following program:

```
tmpdir=mkdir $(mktmp /tmp/XXX)
function cleanup {
  echo "Cleaning $tmpdir"
  rm -rf $tmpdir
}

trap 'cleanup' TERM QUIT INT

echo "I am starting from beginning"
i=0
while true
do
  i=$(($i + 1))
  sleep 10; touch $tmpdir/$i
done
```

What happens if I press control-C when I execute this program?

a. Technically we can bind control-C to generate any signal. Assuming it is default, it will generate SIGINT, which is caught here. The tmdir folder is deleted immediately.
b. After performing the cleanup activity, the program does not stop. It will start immediately after the trap statement – by printing “I am starting from beginning” and proceeding further.
c. After performing cleanup, the program returns the control to where it was when the interrupt came.
d. We exit the program immediately after the cleaning up the tmpdir.

Which of the following statements is true?

1. a and b
2. a and c
3. a and d
4. d alone.
5. None of them are true.

new files list

Given a python file a.py, we want to print out all the python files newer than a.py in a folder and all the subfolders. Consider the following pieces of code:

```
#!/bin/bash

tmpfile=$(mktemp /tmp/rama-XXXX)
find $PWD | grep .py$ > $tmpfile

function get_time {
    mod_time=$(stat -c %y $1)
    mod_secs=$(date -d "$mod_time" +%s)
    echo $mod_secs
}

function clean {
    rm -rf $tmpfile
}

time_stamp=$(get_time "./a.py")
for i in `cat $tmpfile`
do
  [[ $(get_time $i) -gt $time_stamp ]] && echo $i
done

trap 'clean; exit 1' 1 2 3 6
clean; exit 0
```

Consider the following statements:

a. The date command in this program returns the time in seconds since epoch which is used for comparison.
b. This program is overly complex. You can reduce the need for temp file by writing `for i in $(find . | grep ".py$")` and eliminate the need for additional file.
c. The time arithmetic here is complex. You can remove all of it by: `[ "$i" -nt "./a.py" ]` in the loop without having to compute get_time.

Which of the following statements is correct?

1. a alone is correct.
2. a and b are the only correct statements.

3. a and c are the only correct statements.
4. all are correct statements.
5. b and c are corrct statements.

End