Text Areas : PlainText

Possible Answers :

75

Question Number : 241 Question Id : 640653351473 Question Type : SA Calculator : None

Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 2

Question Label : Short Answer Question

The recall of the model with respect to class (Yes) is: ____________

Hint: Enter your answer in %. If your answers is 12%, just enter 12

Response Type : Numeric

Evaluation Required For SA : Yes

Show Word Count : Yes

Answers Type : Equal

Text Areas : PlainText

Possible Answers :

90

System Commands

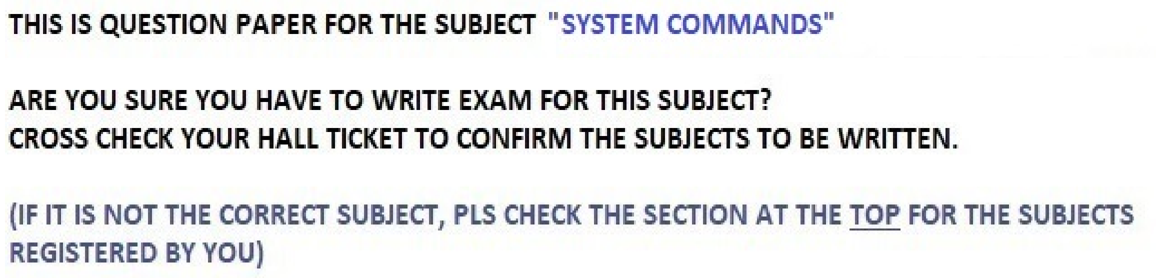
Section Id :	64065322144
Section Number :	14
Section type :	Online
Mandatory or Optional :	Mandatory
Number of Questions :	17
Number of Questions to be attempted :	17
Section Marks :	100
Display Number Panel :	Yes
Group All Questions :	No
Enable Mark as Answered Mark for Review and	Yes

Clear Response :	
Maximum Instruction Time :	0
Sub-Section Number :	1
Sub-Section Id :	64065350446
Question Shuffling Allowed :	No

Question Number : 242 Question Id : 640653351474 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 0

Question Label : Multiple Choice Question



THIS IS QUESTION PAPER FOR THE SUBJECT "SYSTEM COMMANDS"

ARE YOU SURE YOU HAVE TO WRITE EXAM FOR THIS SUBJECT?
CROSS CHECK YOUR HALL TICKET TO CONFIRM THE SUBJECTS TO BE WRITTEN.

(IF IT IS NOT THE CORRECT SUBJECT, PLS CHECK THE SECTION AT THE TOP FOR THE SUBJECTS REGISTERED BY YOU)

Options :

6406531166494. ✓ Yes

6406531166495. ✖ No

Sub-Section Number :	2
Sub-Section Id :	64065350447
Question Shuffling Allowed :	Yes

Question Number : 243 Question Id : 640653351475 Question Type : SA Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 6

Question Label : Short Answer Question

What is the output of the following bash script?

```
n=1234
counter=0
ans=0
while [ $n -gt 0 ]
do
  counter=$(( $n % 10 ))
  ans=$(( $ans * 10 + $counter ))
  n=$(( $n / 10 ))
done
echo $((ans+n))
```

Response Type : Numeric

Evaluation Required For SA : Yes

Show Word Count : Yes

Answers Type : Equal

Text Areas : PlainText

Possible Answers :

4321

Sub-Section Number :	3
Sub-Section Id :	64065350448
Question Shuffling Allowed :	Yes

Question Number : 244 Question Id : 640653351482 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 4

Question Label : Multiple Choice Question

What is the expected output of the following command.

```
find . -type d -name '* *' | wc -l
```

Options :

6406531166526. ✗ Gives the total number of directories in the current working directory.

6406531166527. ✓ Gives the total number of directories with space in their name in the current working directory.

6406531166528. ✖ Gives the total number of files with space in their name in the current working directory.

6406531166529. ✖ Gives the total number of directories with a dot in their name in the current working directory.

Question Number : 245 Question Id : 640653351488 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 4

Question Label : Multiple Choice Question

What command can be used to search for a blank line in a file named `file1`?

Options :

6406531166549. ✖ `$ grep \n file1`

6406531166550. ✖ `$ grep " " file1`

6406531166551. ✔ `$ grep "^$" file1`

6406531166552. ✖ `$ grep "\n" file1`

Sub-Section Number :	4
Sub-Section Id :	64065350449
Question Shuffling Allowed :	Yes

Question Number : 246 Question Id : 640653351476 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 5

Question Label : Multiple Choice Question

The below text is the contents of the file `mycpuinfo`.

```
processor     : 0
vendor_id     : GenuineIntel
cpu family     : 6
model         : 126
model name     : Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz
stepping     : 5
microcode     : 0xb0
cpu MHz         : 1200.000
cache size     : 6144 KB
physical id     : 0
siblings     : 8
core id         : 0
cpu cores     : 4
apicid         : 0
initial apicid     : 0
fpu         : yes
fpu_exception     : yes
cpuid level     : 27
wp         : yes
flags         : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm
constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid
aperfmperf tsc_known_freq pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg
fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave
avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb invpcid_single ssbd ibrs ibpb
stibp ibrs_enhanced tpr_shadow vnmi flexpriority ept vpid ept_ad fsgsbase tsc_adjust
sgx bmi1 avx2 smep bmi2 erms invpcid avx512f avx512dq rdseed adx smap avx512ifma
clflushopt intel_pt avx512cd sha_ni avx512bw avx512vl xsaveopt xsavec xgetbv1 xsaves
split_lock_detect dtherm ida arat pln pts hwp hwp_notify hwp_act_window hwp_epp
hwp_pkg_req avx512vbmi umip pku ospke avx512_vbmi2 gfni vaes vpclmulqdq avx512_vnni
avx512_bitalg avx512_vpopcntdq rdpid sgx_lc fsrm md_clear flush_l1d arch_capabilities
vmx flags     : vnmi preemption_timer posted_intr invvpid ept_x_only ept_ad ept_1gb
flexpriority apicv tsc_offset vtpr mtf vapic ept vpid unrestricted_guest vapic_reg vid
ple pml ept_mode_based_exec tsc_scaling
bugs         : spectre_v1 spectre_v2 spec_store_bypass swapgs itlb_multihit srbds
mmio_stale_data
bogomips     : 2380.80
clflush size     : 64
cache_alignment     : 64
address sizes     : 39 bits physical, 48 bits virtual
power management:
```

Select the command that will print the number of CPU cores on the system. The number of CPU cores is given in the text as a value to the key "cpu cores". Thus, your output should be "4".

Note: Use of `grep` command's option:

```
-o, --only-matching
    Print only the matched (non-empty) parts of a matching
    line, with each such part on a separate output line.
```

Options :

6406531166497. ✖

```
grep cpu mycpuinfo
```

6406531166498. ✖

```
grep -o "cpu cores" mycpuinfo
```

6406531166499. ✓

```
grep "cpu cores" mycpuinfo | egrep -o "[[:digit:]]+"
```

6406531166500. ✗

```
grep -o "cpu cores" mycpuinfo | egrep -o "[[:digit:]]+"
```

Question Number : 247 Question Id : 640653351477 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 5

Question Label : Multiple Choice Question

What is the output of the following script if the input to stdin is `45.53`?

```
read var

function operate()
{
  temp=$1
  temp=${temp%.*}
  echo $temp
}

echo $( operate $var )
```

Options :

6406531166501. ✗ 0

6406531166502. ✗ 53

6406531166503. ✓ 45

6406531166504. ✗ 1

6406531166505. ✗ 45.53

Question Number : 248 Question Id : 640653351479 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 5

Question Label : Multiple Choice Question

```
sed -n '/[[:digit:]]\{3\}/ p' myfile
```

What does the above command do?

Note: Use of `sed` command option:

```
-n, --quiet, --silent
  suppress automatic printing of pattern space,i.e. print only the matched lines.
```

Options :

6406531166512. ✗ Prints the line having only three digits.

6406531166513. ✗ Prints the line having at least three digits.

6406531166514. ✓ Prints the line having three consecutive digits.

6406531166515. ✗ Prints the line having at most two consecutive digits.

Question Number : 249 Question Id : 640653351494 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 5

Question Label : Multiple Choice Question

When the command `ls -l` is run on the current directory, the output obtained is

```
-rw-r--r--  1 user  group   0 Nov 30 11:08 rand1.txt
-rw-r--r--  1 user  group   0 Nov 29 11:08 rand2.txt
-rw-r--r--  1 user  group   0 Nov 29 11:08 rand3.md
-rw-r--r--  1 user  group   0 Nov 28 11:08 rand4.awk
-rwxr-xr-x 1 user  group   0 Nov 10 14:03 script.sh
-rwxr-xr-x 1 user  group  1 Nov 30 20:44 test.sh
```

What is the correct output format for this bash script?

```
for line in `ls`; do
        details=`ls -l $line`
        echo $line: ${details:4:6}
done
```

Options :

6406531166575. ✖

```
rand1.txt: {details:4:6}
rand2.txt: {details:4:6}
rand3.txt: {details:4:6}
rand4.txt: {details:4:6}
script.sh: {details:4:6}
test.sh: {details:4:6}
```

6406531166576. ✖

```
rand1.txt: r-
rand2.txt: r-
rand3.txt: r-
rand4.txt: r-
script.sh: r-
test.sh: r-
```

6406531166577. ✖

```
rand1.txt: -r
rand2.txt: -r
rand3.txt: -r
rand4.txt: -r
script.sh: xr
test.sh: xr
```

6406531166578. ✔

```
rand1.txt: r--r--
rand2.txt: r--r--
rand3.txt: r--r--
rand4.txt: r--r--
script.sh: r-xr-x
test.sh: r-xr-x
```

6406531166579. ✖

```
rand1.txt: -r--r-
rand2.txt: -r--r-
rand3.txt: -r--r-
rand4.txt: -r--r-
script.sh: xr-xr-
test.sh: xr-xr-
```

Sub-Section Number : 5

Sub-Section Id : 64065350450

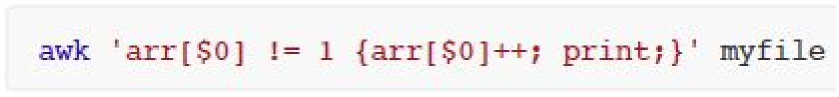
Question Shuffling Allowed : Yes

Question Number : 250 Question Id : 640653351480 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 6

Question Label : Multiple Choice Question

What will the below command print?



```
awk 'arr[$0] != 1 {arr[$0]++; print;}' myfile
```

Options :

6406531166516. ✖ Second occurrences of duplicate lines.

6406531166517. ✖ Distinct lines in the alphabetically sorted order.

6406531166518. ✔ Distinct lines in the order of first occurrence.

6406531166519. ✖ The lines that are present more than once.

Question Number : 251 Question Id : 640653351483 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 6

Question Label : Multiple Choice Question

Here is a part of information from AWK manual,

```
gsub(r, s [, t])  For each substring matching the regular  expres-
                  sion r in the string t, substitute the string s,
                  and return the number of substitutions.  If t is
                  not  supplied,  use $0.  An & in the replacement
                  text is replaced with the text that was actually
                  matched.  Use \& to get a literal &.  (This must
                  be typed as "\\&";
```

The contents of the file `myfile` are given below

```
Ram
Laila
Ahmed
Ragav
Peter
```

What will be the output after running the below command?

```
awk '{
    gsub(/.*/, NR":&");
    print $0;
}' myfile
```

Options :

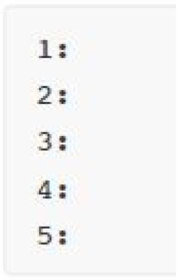
6406531166530. ✓

```
1:Ram
2:Laila
3:Ahmed
4:Ragav
5:Peter
```

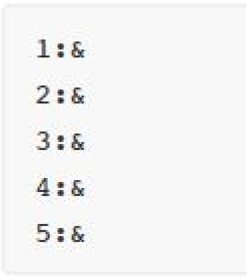
6406531166531. ✗

```
NR:Ram
NR:Laila
NR:Ahmed
NR:Ragav
NR:Peter
```

6406531166532. ✗



```
1:
2:
3:
4:
5:
```

6406531166533. 

```
1:&
2:&
3:&
4:&
5:&
```

Sub-Section Number : 6

Sub-Section Id : 64065350451

Question Shuffling Allowed : Yes

Question Number : 252 Question Id : 640653351478 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 7

Question Label : Multiple Select Question

Consider a regular list as shown in the sample input, that is stored in a single line in the file `mylist`.

Select the sed script to pretty print this list such that in the output the first and the last line have the starting and the ending brackets, and the elements of the list are printed between the first and the last lines, one element on each line indented by a tab. The elements should be printed in the same order from top to bottom as they appear in the list from left to right.

Note: The tab and newline characters are specified by `\t` and `\n` respectively.

Sample Input

```
[1,2,3,4]
```

Sample Output

```
[
	1,
	2,
	3,
	4
]
```

Options :

6406531166506. ✓

```
sed 's/\[/\[\n\t/g' mylist |
sed 's/\]/\n]/' |
sed 's/,/,\n\t/g'
```

6406531166507. ✗

```
sed 's/\[/\[\t\n/g' mylist |
sed 's/\]/\n]/' |
sed '/^[[:blank:]]/ s/,/,\t\n/'
```

6406531166508. ✓

```
sed 's/\[/\[\n\t/g' mylist |
sed 's/\]/\n]/' |
sed '/^[[:blank:]]/ s/,/,\n\t/g'
```

6406531166509. ✗

```
sed 's/\[/\[\n\t/g' mylist |
sed 's/\]/\n]/' |
sed '/^[[:blank:]]/ s/,/,\n\t/'
```

6406531166510. ✗

```
sed 's/\[/\[\t\n/g' mylist |
sed 's/\]/\n]/' |
sed '/^[[:blank:]][[:blank:]]*/ s/,/,\t\n/'
```

6406531166511. ✗

```
sed 's/\[/\[\t\n/g' mylist |
sed 's/\]/\n]/' |
sed 's/,/,\t\n/g'
```

Sub-Section Number : 7

Sub-Section Id : 64065350452

Question Shuffling Allowed : Yes

Question Number : 253 Question Id : 640653351481 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 6

Question Label : Multiple Select Question

Select all the commands that prints the from line `4` to line `7` of file `myfile` (line 4 and 7 incluuded).
Assume that the `myfile` contains at least 7 lines.

Options :

6406531166520. ✓

```
head -7 myfile | tail -4
```

6406531166521. ✗

```
tail -4 myfile | tail -7
```

6406531166522. ✓

```
sed -n '4,7 p' myfile
```

6406531166523. ✗

```
sed -n '4-7 p' myfile
```

6406531166524. ✓

```
awk 'NR >= 4 && NR <= 7 {print;}' myfile
```

6406531166525. ✓

```
awk '{if(NR>=4 && NR <=7){print;}}' myfile
```

Question Number : 254 Question Id : 640653351487 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 6

Question Label : Multiple Select Question

Consider a file `/home/user/script` that exists currently on the system. Below commands are run to create links.

```
mkdir /home/user/links/
ln /home/user/script /home/user/links/script_link
ln /home/user/links/script_link /home/user/links/script_link_link
```

Which of the following statements are **true** ?

Options :

6406531166544. ✓ The link `script_link` will be accessible even if it is moved to different directory within the file system.

6406531166545. ✓ Files `script_link` and `script` will have the same inode number.

6406531166546. ✖ If we create a copy of the file `script`, the copy will also be linked automatically from the file `script_link`.

6406531166547. ✖ If the file `script_link` is deleted, the file `script_link_link` will be inaccessible.

6406531166548. ✗ If the files `script` and `script_link` both are deleted, the file `script_link_link` will be inaccessible.

Question Number : 255 Question Id : 640653351489 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 6

Question Label : Multiple Select Question

The following bash script performs some operation. This script is run with some command line arguments. The command line argument values contains only alphanumeric characters.

```
#!bin/bash
args=0
for i in "$@"; do
  args=$((args+1))
done;
echo $args
```

Which of the following bash scripts is an will give the same output as the above script?

Options :

6406531166553. ✗
```
#!bin/bash
echo $$
```

6406531166554. ✓
```
#!bin/bash
echo $#
```

6406531166555. ✓
```
#!bin/bash
echo $@ | awk 'END{print NF}'
```

6406531166556. ✓

```
#!bin/bash
echo $@ | wc -w
```

Question Number : 256 Question Id : 640653351490 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 6

Question Label : Multiple Select Question

Which of the following commands will print the lines that do not start with `#`, `'` or `//`.

Hint: `grep` / `egrep` command option use:

```
-v, --invert-match
              Invert the sense of matching, to select non-matching
              lines.
```

Options :

6406531166557. ✗ `egrep  -v "^#|^'|^\//" code.txt`

6406531166558. ✓ `grep  -v "^#\|^'\|^\/\/" code.txt`

6406531166559. ✓ `egrep  -v "^#|^'|^\/\/" code.txt`

6406531166560. ✗ `grep  -v "^#\|^'\|^\//" code.txt`

6406531166561. ✗ `egrep  -v "^#|^'|^//" code.txt`

6406531166562. ✗ `grep  -v "^#\|^'\|^//" code.txt`

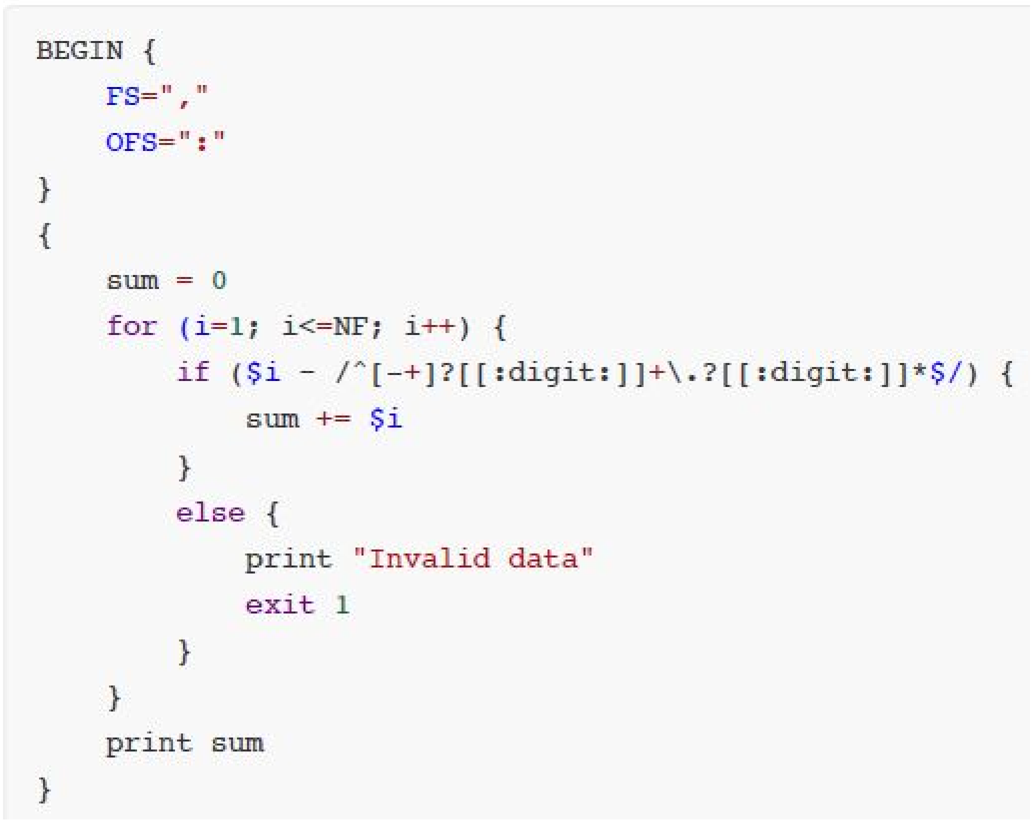
Sub-Section Number :	8
Sub-Section Id :	64065350453
Question Shuffling Allowed :	No

Question Id : 640653351484 Question Type : COMPREHENSION Sub Question Shuffling Allowed : No Group Comprehension Questions : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Question Numbers : (257 to 258)

Question Label : Comprehension

Consider the AWK script and answer the given subquestions



```
BEGIN {
    FS=","
    OFS=":"
}
{
    sum = 0
    for (i=1; i<=NF; i++) {
        if ($i ~ /^[-+]?[[:digit:]]+\.?[[:digit:]]*$/) {
            sum += $i
        }
        else {
            print "Invalid data"
            exit 1
        }
    }
    print sum
}
```

Sub questions

Question Number : 257 Question Id : 640653351485 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 4

Question Label : Multiple Select Question

Select all the correct statement(s) with respect to the given AWK script.

Options :

6406531166534. ✓ The fields in the input to the script are comma(,) separated.

6406531166535. ✗ The number of field can be at most `5`.

6406531166536. ✓ If any of the field values is `.123` then `Invalid data` will be printed.

6406531166537. ✓ If any of the field values is `+-1.123` then `Invalid data` will be printed.

6406531166538. ✓ if any of the fields contain an alphabet then `Invalid data` will be printed.

6406531166539. ✗ The fields in the input to the script are colon(`:`) separated.

Question Number : 258 Question Id : 640653351486 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 5

Question Label : Multiple Choice Question

Select the output of the AWK script on the input given below.

```
1,2,3
1.1,2.1,3.1
-1.1,2.1,3.1
+1.1,2.1,3.1
a,b,2
.1,89,1
```

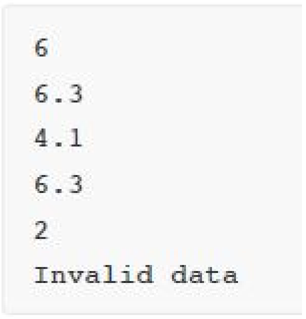
Options :

6406531166540. ✗

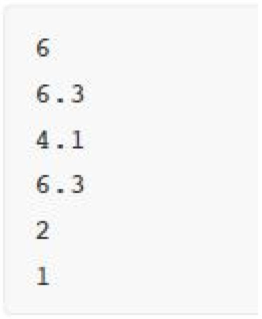
```
Invalid data
```

6406531166541. ✓

```
6
6.3
4.1
6.3
Invalid data
```

6406531166542. ✖ 

```
6
6.3
4.1
6.3
2
Invalid data
```

6406531166543. ✖ 

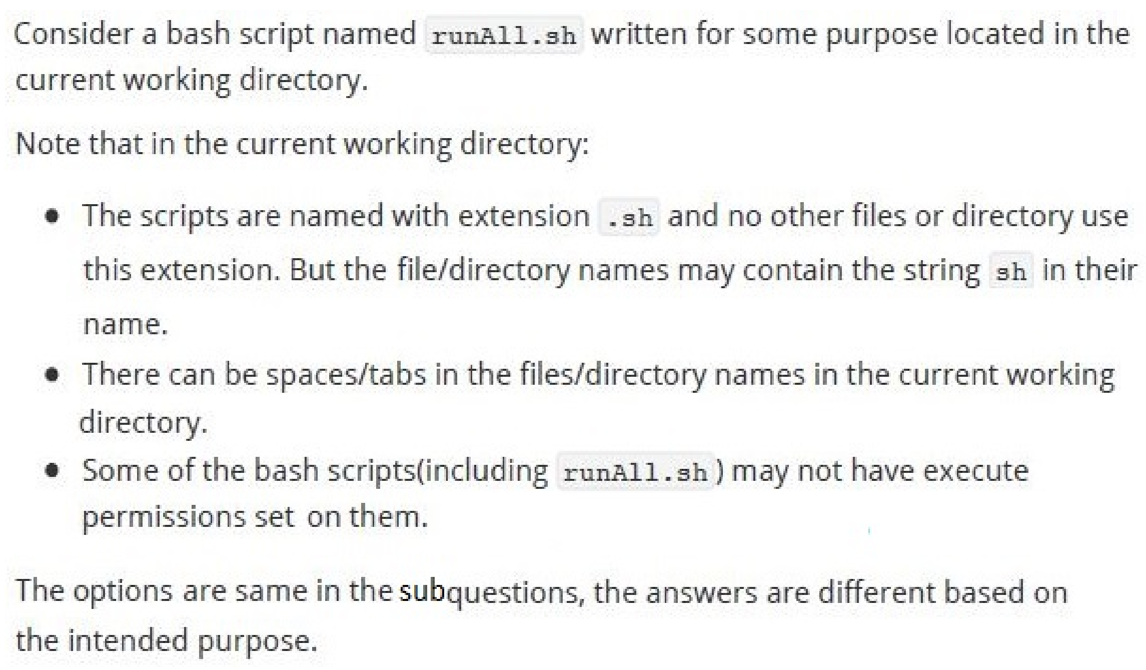
```
6
6.3
4.1
6.3
2
1
```

Sub-Section Number :	9
Sub-Section Id :	64065350454
Question Shuffling Allowed :	No

Question Id : 640653351491 Question Type : COMPREHENSION Sub Question Shuffling Allowed : No Group Comprehension Questions : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Question Numbers : (259 to 260)

Question Label : Comprehension


Consider a bash script named `runAll.sh` written for some purpose located in the current working directory.

Note that in the current working directory:

- The scripts are named with extension `.sh` and no other files or directory use this extension. But the file/directory names may contain the string `sh` in their name.
- There can be spaces/tabs in the files/directory names in the current working directory.
- Some of the bash scripts(including `runAll.sh`) may not have execute permissions set on them.

The options are same in the subquestions, the answers are different based on the intended purpose.

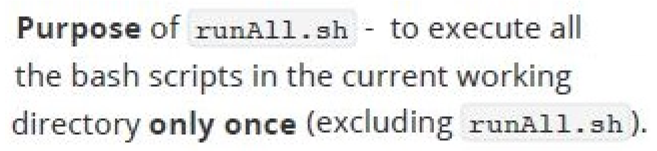
Based on the above data, answer the given subquestions

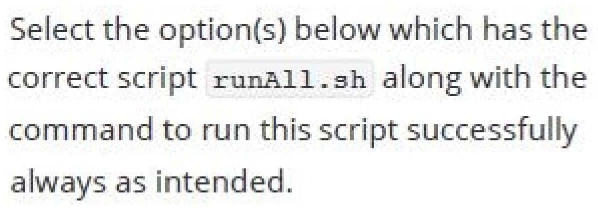
Sub questions

Question Number : 259 Question Id : 640653351492 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 8

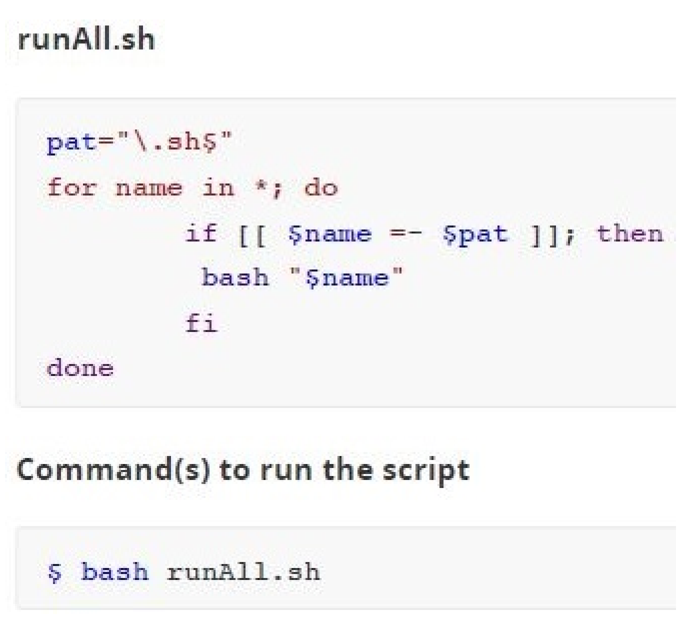
Question Label : Multiple Select Question


Purpose of `runAll.sh` - to execute all the bash scripts in the current working directory **only once** (excluding `runAll.sh`).


Select the option(s) below which has the correct script `runAll.sh` along with the command to run this script successfully always as intended.

Options :

6406531166563. ✖

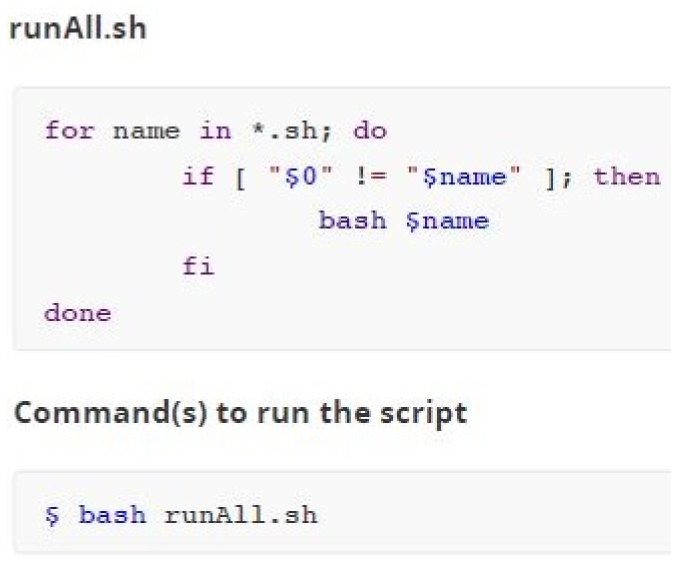


runAll.sh

```
pat="\.sh$"
for name in *; do
        if [[ $name =~ $pat ]]; then
         bash "$name"
        fi
done
```

Command(s) to run the script

```
$ bash runAll.sh
```



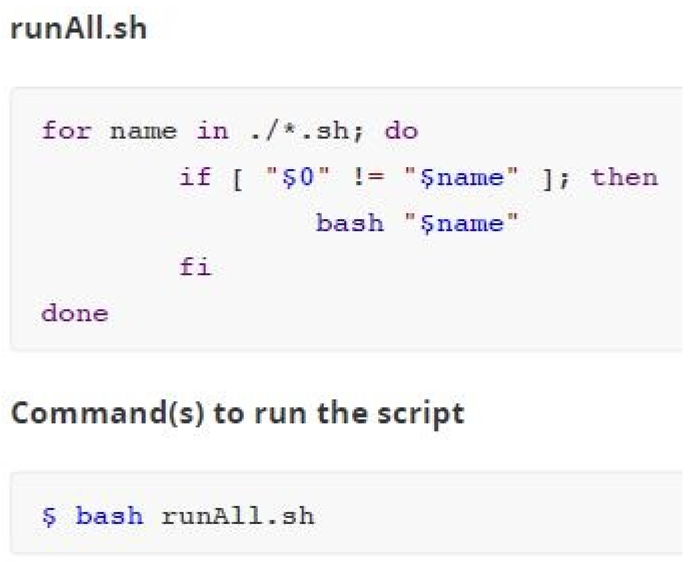
runAll.sh

```
for name in *.sh; do
        if [ "$0" != "$name" ]; then
                bash $name
        fi
done
```

Command(s) to run the script

```
$ bash runAll.sh
```

6406531166564. ✗



runAll.sh

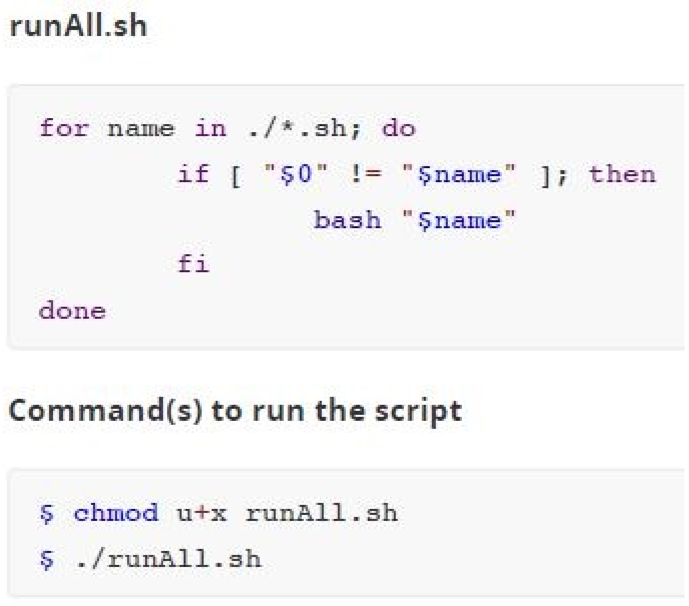
```
for name in ./*.sh; do
        if [ "$0" != "$name" ]; then
                bash "$name"
        fi
done
```

Command(s) to run the script

```
$ bash runAll.sh
```

6406531166565. ✗

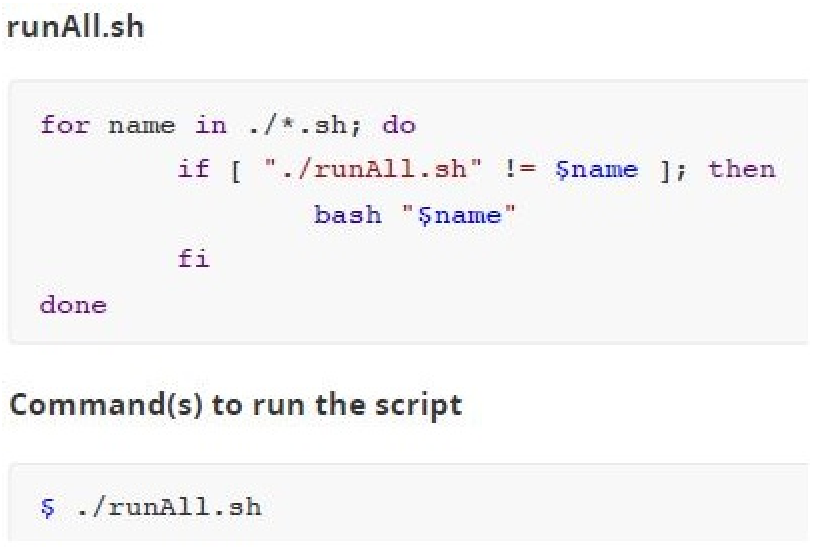
6406531166566. ✓


runAll.sh

```
for name in ./*.sh; do
        if [ "$0" != "$name" ]; then
                bash "$name"
        fi
done
```

Command(s) to run the script

```
$ chmod u+x runAll.sh
$ ./runAll.sh
```

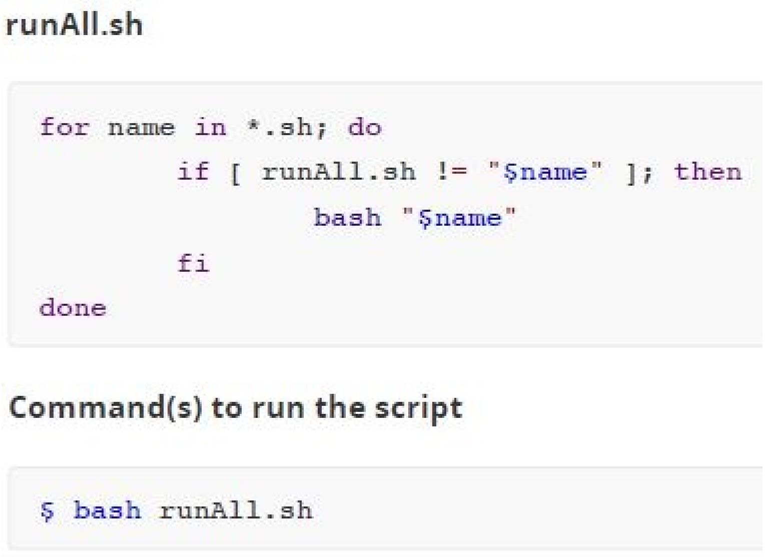

runAll.sh

```
for name in ./*.sh; do
        if [ "./runAll.sh" != $name ]; then
                bash "$name"
        fi
done
```

Command(s) to run the script

```
$ ./runAll.sh
```

6406531166567. ✖


runAll.sh

```
for name in *.sh; do
        if [ runAll.sh != "$name" ]; then
                bash "$name"
        fi
done
```

Command(s) to run the script

```
$ bash runAll.sh
```

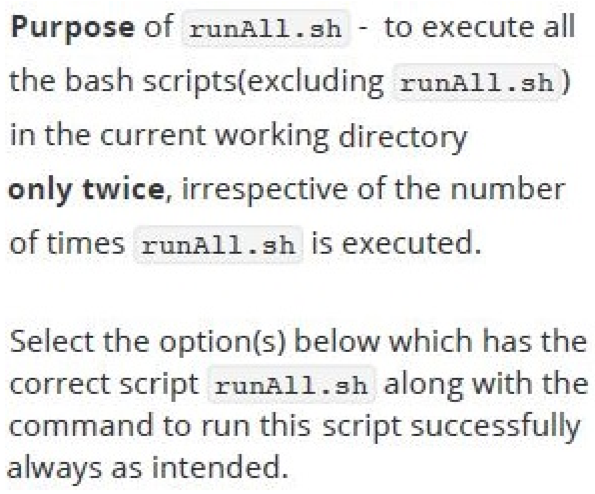
6406531166568. ✔

Question Number : 260 Question Id : 640653351493 Question Type : MCQ Is Question

Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 6

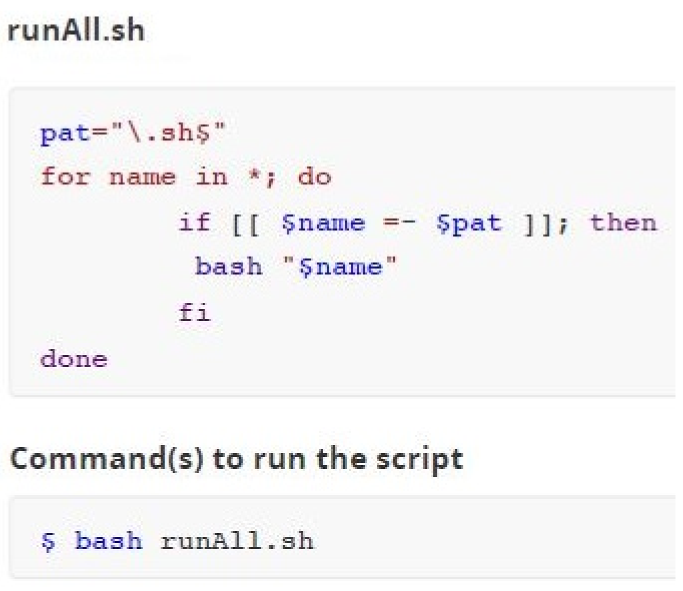
Question Label : Multiple Choice Question



Purpose of `runAll.sh` - to execute all the bash scripts(excluding `runAll.sh`) in the current working directory **only twice**, irrespective of the number of times `runAll.sh` is executed.

Select the option(s) below which has the correct script `runAll.sh` along with the command to run this script successfully always as intended.

Options :



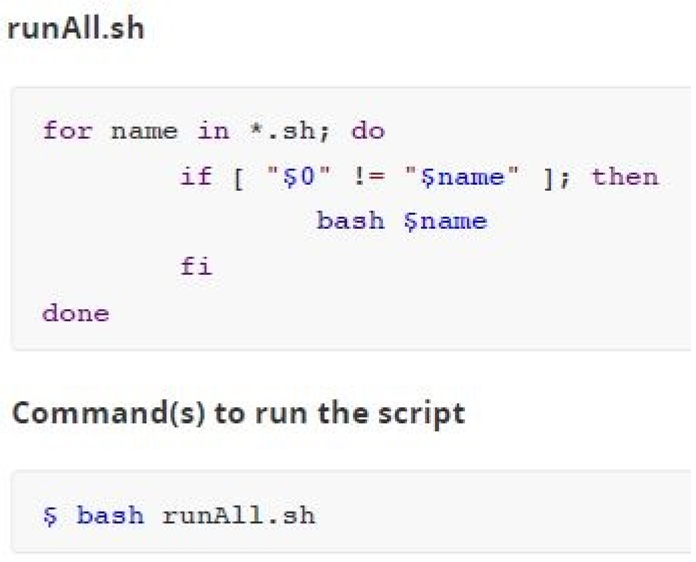
runAll.sh

```
pat="\.sh$"
for name in *; do
        if [[ $name =~ $pat ]]; then
         bash "$name"
        fi
done
```

Command(s) to run the script

```
$ bash runAll.sh
```

6406531166569. ✖



runAll.sh

```
for name in *.sh; do
        if [ "$0" != "$name" ]; then
                bash $name
        fi
done
```

Command(s) to run the script

```
$ bash runAll.sh
```

6406531166570. ✖

runAll.sh

```
for name in ./*.sh; do
        if [ "$0" != "$name" ]; then
                bash "$name"
        fi
done
```

Command(s) to run the script

```
$ bash runAll.sh
```

6406531166571. ✓

runAll.sh

```
for name in ./*.sh; do
        if [ "$0" != "$name" ]; then
                bash "$name"
        fi
done
```

Command(s) to run the script

```
$ chmod u+x runAll.sh
$ ./runAll.sh
```

6406531166572. ✖

runAll.sh

```
for name in *.sh; do
        if [ runAll.sh != $name ]; then
                bash "$name"
        fi
done
```

Command(s) to run the script

```
$ chmod u+x runAll.sh
$ ./runAll.sh
```

6406531166573. ✖

6406531166574. ✖

runAll.sh

```
for name in *.sh; do
        if [ runAll.sh != "$name" ]; then
                bash "$name"
        fi
done
```

Command(s) to run the script

```
$ bash runAll.sh
```