Bash Programming

TOTAL POINTS 10

1. Which of the following are requirements for variable names? 1 point

- [ ] Every character in the name is uppercase.
- [ ] Numbers are not allowed in variable names.
- [x] The variable name starts with a letter.
- [x] Every character in the name is lowercase.

2. What does the exit status of a program indicate? 1 point

- ( ) The exit status of a program indicates how a program will be completed once it's exhausted all of its code.
- ( ) The exit status of a program indicates the value that was computed by the program.
- (•) The exit status of a program indicates whether the program was executed successfully or whether an error occurred.
- ( ) The exit status of a program indicates how many programs were running at the same time as a particular program.

3. What is printed to the console by the following command? 1 point

```
echo Demetrius || [[ 6 -eq 7 ]] || echo Helena && echo Hermia || [[ 7 -gt 4 ]]
```

- ( ) Nothing is printed to the console.
- ( )
```
Helena
Hermia
```
- (•)
```
Demetrius
Hermia
```
- ( )
```
Demetrius
Helena
```

4. Consider the following program called numrange.sh: 1 point

```
#!/usr/bin/env bash
# File: numrange.sh

odd=$(echo "$1 % 2" | bc)

if [[ $odd -eq 0 ]]
then
  status="even"
else
  status="odd"
fi

if [[ $1 -gt 0 ]] && [[ $1 -lt 10 ]]
then
  location="in"
else
  location="out of"
fi

echo "This number is $status and $location range."
```

Which of the following is the result of commands below?

```
bash numrange.sh 6
bash numrange.sh 11
bash numrange.sh 400 10
```

- (•)
```
This number is even and in range.
This number is odd and out of range.
This number is even and out of range.
```

```
This number is even and in range.
This number is odd and out of range.
- error - too many arguments
```

```
This number is odd and out of range.
This number is even and in range.
This number is even and out of range.
```

```
This number is even and out of range.
This number is odd and in range.
This number is even and out of range.
```

5. What is the result of the script below? 1 point

```
lab=(jeff roger brian)
lab[3]=sean
lab=("${lab[*]}" "${lab[*]}")
echo ${#lab[*]}
```

- (•) 2
- ( ) 1
- ( ) 6
- ( ) 9

6. Consider the following program called reqseq.sh: 1 point

```
#!/usr/bin/env bash
# File: repseq.sh

sequence=$(eval echo {$1..$2})

for i in $sequence
do
  compute=$(echo "$i % $3" | bc)
  result="$result $compute"
done

echo $result
```

Which of the commands below would create the following output?

```
1 2 0 1 2 0 1 2 0
```

- ( )
```
bash repseq.sh 1 6 2
```
- ( )
```
bash repseq.sh 1 9 2
```
- ( )
```
bash repseq.sh 1 6 3
```
- (•)
```
bash repseq.sh 1 9 3
```

7. What's the purpose of the **local** keyword? 1 point

- ( ) The **local** keyword allows you to create a function such that the function can be used within your shell the same way you would use a command.
- ( ) The **local** keyword ensures that all of the actions taken by a particular function do not affect the global computing environment.

- (●) The **local** keyword allows you to assign the value of a variable within a function without changing the global value of that variable.
- ( ) The **local** keyword stores the value of several variables locally so that they can be accessed later on within a script.

8. Which of the following are **not** part of the Unix Philosophy? 1 point

- [ ] A program a should do one thing well.
- [ ] Programs should be composable.
- [ ] Programs should have easy to understand error messages.
- [x] Programs should be quiet.
- [x] A program should run quickly.

9. What actions are taken by the following commands? 1 point

```
1 chmod a+x my_program
2 chmod go-rw my_program
```

- ( ) 1. Allows only the owner to execute **my_program**.
  2. Prevents anyone other than the owner from reading or modifying **my_program**.
- ( ) 1. Allows anyone to execute **my_program**.
  2. Prevents the owner from reading or modifying **my_program**.
- (●) 1. Allows anyone to execute **my_program**.
  2. Prevents anyone other than the owner from reading or modifying **my_program**.
- ( ) 1. Allows only the owner to execute **my_program**.
  2. Prevents the owner from reading or modifying my_program.

10. What is one reason you might want to modify the **PATH** environmental variable? 1 point

- ( ) Modifying the PATH makes it easier to switch between programs when you are using multiple shells at once.
- ( ) You can make functions available to you on the command line by including the PATH variable inside of the definition of a function.
- (●) You can add a directory containing your own programs to the PATH which allows you to access them on the command line.
- ( ) The PATH can be modified in the bash profile which is where aliases are defined. The bash profile is run every time you start a shell.

[ ] I, **Piyush Sambhi**, understand that submitting work that isn't my own may result in permanent failure of this course or deactivation of my Coursera account.

Learn more about Coursera's Honor Code

Save Submit