1. What is a resource type in Puppet? Give three examples of resource types.

A resource type defines the range of characteristics (parameters) of a resource,e.g. a user, group, file, service, package, etc, list them all with `puppet describe--list`

2. What is the Puppet Resource Abstraction Layer (RAL)?

It is Puppets way of abstracting away the details of implementation on eachplatform, e.g. you do not have to worry about whether packages should be in-stalled with apt or yum, puppet figures that our for you.

3. You can get the ip (v4) address of a host with `facter ipaddress`, but how do you access this same information as a top-scope variable in a Puppet manifest?

POSSIBLE SOLUTION:

```
${::ipaddress}
```

4. Given that all services your team is responsible for are containerized (run in containers), how does this simplify server management?

You can simplify your server configuration, you do not have to prepare an environment with e.g. different versions of the same libraries. Simpler config is faster to apply and easier to secure. You focus your config on the basics with only the addition of container runtime config.

5. Write a Puppet manifest that ensures your home directory (`/home/ubuntu`) is only accessible by you (permission `0750`).

POSSIBLE SOLUTION:

```
file { '/home/ubuntu':
      mode => '0750',
}
```

6. Write a Puppet manifest that ensures `/home/ubuntu/public_html` is a symbolic link to `/var/www/ubuntu`. Is `/var/www/ubuntu` created if it does not exist?

POSSIBLE SOLUTION:

```
file { '/home/ubuntu/public_html':
      ensure => link,
      target => '/var/www/ubuntu'
}

# No, /var/www/ubuntu is not created,
# Puppet only does what you ask it to do
```

7. Write a Puppet manifest that ensures that the packages `vim`, `nano`, `jed` and `jove` are installed. Use an array to store the list of packages so you only need to declare the `package` resource type once.

POSSIBLE SOLUTION:

```
$texteditors = [ "vim", "nano", "jed", "jove" ]
package { $texteditors:
      ensure => 'present',
}
```

8. Write a Puppet manifest that ensures the user `data` exists with encrypted password `$6$38Tdw...` and belonging to the `sudo` group. Also make sure the group `data` exists before ensuring the user `data` exists.

POSSIBLE SOLUTION:

```
group { 'data':
      ensure => present,
}

user { 'data':
      ensure => present,
# the following two lines should be one (no line break):
      password => '$6$38Tdw8HdN5eHALGh$nC5X8ggGk0AzvR6k5WjBJxcCqk3nbVFe6u
      vupd3CXYjF3Eo0Ltifk5SST6KWFEYi59wzlbwto0MtIeQv95ouw0',
      gid => 'data',
      home => '/home/data',
      managehome => true,
      shell => '/bin/bash',
      groups => 'sudo',
      require => Group['data'],
}
```

9. Write a Puppet manifest which does the following in sequence: (a) ensure the ssh package is in its latest version. (b) edit the config file `sshd_config`, use the following resource declaration for this:

```
augeas { 'sshd_config':
      context => '/files/etc/ssh/sshd_config',
      changes => 'set X11Forwarding yes',
}
```

(c) make sure that the ssh service is running and restarted if its configuration has been changed.

POSSIBLE SOLUTION:

```
package {'ssh':
      ensure => latest,
      before => Augeas['sshd_config'],
}

augeas { 'sshd_config':
      context => '/files/etc/ssh/sshd_config',
      changes => 'set X11Forwarding yes',
      }

service { 'ssh':
      ensure => running,
      subscribe => Augeas['sshd_config'],
}
```

10. Write a Puppet manifest that makes sure the software package `mybackup` is installed in its latest version, and creates a cron entry to run the command `mybackup /home/data` as root every half hour. The manifest must include a notify relationship between the package and cron resource declarations.
Note. You will not be able to execute this task as there is no real package called “mybackup”.

POSSIBLE SOLUTION:

```
package { 'mybackup':
      ensure => 'latest',
}

cron { 'rand' :
      command => "0,30 * * * * root mybackup /home/data",
      user => root,
      subscribe => Package['mybackup'],
}
```