PERL LAB 6 - Lists and Hashes

1. Enter the following script into **vi**, call it **coord.pl**, run it and convince youself that you know what it is doing:

```
@coord_3D = (100, 200, -200);

print "@coord_3D\n";

($x, $y, $z) = @coord_3D;

print '$x = ' . $x . ', $y = ' . $y . ', $z = ' . $z, "\n";

$x /= 2;
$y /= 2;
$z /= 2;

print '$x = ' . $x . ', $y = ' . $y . ', $z = ' . $z, "\n";

@coord_3D = ($x, $y, $z);
print "@coord_3D\n";
```

2. Run the following script, call it **mytime.pl**:

```
($sec, $min, $hour, $month_day, $month, $year, $week_day, $year_day, $summer_time) = gmtime;
++$month;
$year += 1900;  # Look: no Y2K bugs here!
print "The date is: $month_day/$month/$year\n";
```

3. Run the following script, call it **primes.pl**:

```
sub the_first_primes {
     @primes = (1, 2, 3, 5, 7, 11, 13, 17, 19, 23);

     return wantarray ? @primes : ($#primes+1);
}

@list = the_first_primes;

$scalar = the_first_primes;

print "The subroutine returned the list value:   @list\n";
print "The subroutine returned the scalar value: $scalar\n";
```

4. Run this script, call it **bashers.pl**, against the **/etc/passwd** file:

```
@bash_users = map { m[(^\w+).*/bin/bash$] ; $1 } <>;

print "Bashers on this system are:\n";

foreach $b (@bash_users)
{
     print "  User = ", $b, "\n" if defined( $b );
}
```

5. Run this script, call it **mycp.pl**:

```
sub my_copy {
     my $copy_one = shift @_;   # Note: the @_ is optional;
     $copy_one = "one has been changed";
     $_[0] = "two has been changed";  # Why zero?
}

$one = "this is one";
$two = "this is two";

my_copy ($one, $two);

print '$one = ' . $one, "\n";
print '$two = ' . $two, "\n";
```

6. Run this script, call it **myset.pl**. Compare the output from the script to the LINUX **set** command:

```
#!/usr/bin/perl -w

foreach $var (sort keys %ENV)
{
     print "$var = $ENV{$var}\n";
}
```

7. Run the LINUX **who** command, then run this script, call my **mywho.pl**, and compare the results:

```
#!/usr/bin/perl -w


%unique = ();  # The hash is initially empty.

# On the next line, the use of `who` allows Perl to call an
# operating system command and have the results delivered as
# lined input to the script.

for (`who`)
{
     s/\s.*\n//;          # Remove unwanted space.
     $unique{$_}++;  # Update the hash with $_ (current thing).
}

@users = sort keys %unique;    # Produce a sorted list.

print "The logged in users are: @users\n";
```