Informe de Laboratorio 06

Tema: Perl

Estudiante	Escuela	Asignatura
Luis Gustavo Sequeiros Condori lsequeiros@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web I Semestre: II

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 16 Diciembre 2023	Al 23 Diciembre 2023

1. Temas a Tratar:

- Ejercicios en Perl

2. Actividades

2.1. Ejercicio 1

Escriba un programa en Perl que le pida su nombre y lo salude usando un mensaje con el nombre ingresado.

Listing 1: Programa llamado *ej1.pl*

```perl
#!/usr/bin/perl

$name = <STDIN>;
chomp($name);
print "Hello $name!\n";
```

Utilizo la entrada estándar, sin embargo, lee la entrada con un salto de línea. Utilizo la subrutina *chomp* para eliminar ese salto de línea. Luego, simplemente imprimo el saludo con el nombre ingresado.

2.2. Ejercicio 2

Programe la función mayor que reciba un arreglo de números y devuelva el mayor de todos. int mayor(int[] a). Sólo puede usar ciclos y condicionales, no podrá usar otras funciones del lenguaje Perl.

Listing 2: Programa llamado *ej2.pl*

```perl
#!/usr/bin/perl

sub Mayor {
  $may = $_[0];
  foreach $var(@_) {
```

```
6       if($var > $may) {
7         $may = $var;
8       }
9     }
10    return $may;
11  }
12
13  @arr = (1,2,10,4,5);
14  $may = Mayor(@arr);
15  print "The mayor number is: $may\n";
```

Se crea la subrutina *Mayor*, el cual itera en el arreglo que se ingresa como parámetro, se toma como elemento mayor al primer elemento en el arreglo, si hay otro elemento mayor, pues se cambia. Retorna el número mayor en el arreglo.

2.3. Ejercicio 3

Escriba una función que reciba un string y retorne el mismo string invertido. Por ejemplo si la función recibe roma", deberá devolver .ªmor". Sólo puede usar ciclos y condicionales, no podrá usar otras funciones del lenguaje Perl.

Listing 3: Programa llamado *ej3.pl*

```
1   #!/usr/bin/perl
2
3   sub Invert {
4     $str = "";
5     for($i = 0; $i < length($_[0]); $i++){
6       $char = substr($_[0], length($_[0]) - $i - 1, 1);
7       $str = $str.$char;
8     }
9     return $str;
10  }
11  $input = <STDIN>;
12  chomp($input);
13  $invert = Invert($input);
14  print $invert."\n";
```

La subrutina *Invert* itera en cada caracter del string ingresado como parámetro, emepzando por el final. Luego, ese caracter se concatena a un nuevo string. Al final obtengo el string invertido.

2.4. Ejercicio 4

Estamos al final del semestre y para calcular su promedio el profesor del curso desea eliminar la peor nota y duplicar la mayor nota. De este modo, si sus notas fueran 12, 15, 17 y 14; el profesor eliminaría el 12 y duplicaría el 17, entonces sus notas serían 17, 15, 17 y 14 y sobre ellas calcularía el promedio. Usted deberá programar una función que reciba las notas y devuelva el promedio, según se explicó y para una cantidad de notas no determinada. Para programar su función no podrá usar ningún tipo de condicionales, sólo las funciones max y min de perl.

Listing 4: Programa llamado *ej4.pl*

```
1   #!/usr/bin/perl
2   use List::Util qw(max min);
3   sub Average {
```

```
  $mayor = max(@_);
  $minor = min(@_);
  $sum = 0;
  foreach $var (@_){
    $sum += $var;
  }
  $av = ($sum - $minor + $mayor) / @_;
  return $av;
}
$ave = Average(10,10,20);
print $ave."\n";
```

Se utiliza el módulo *List::Util* para poder utilizar las subrutinas *max y min*. En la subrutina *Average*, se extrae el elemento mayor y menor en el arreglo de parámetros. Se suman todos los elementos del arreglo y luego, para hallar el promedio de acuerdo a las condiciones dadas, se resta el menor y se suma el mayor, luego se divide entre el número de elementos del arreglo.

2.5. Ejercicio 5

Se define una celebridad como una persona que no conoce a nadie, pero que todos la conocen. Se dispone de una matriz M, donde la posición M[i,j] es True si la persona i, conoce a la persona j; si la persona i, no conoce a la persona j, entonces M[i,j] será False. Describa un algoritmo que dada una matriz M, logre determinar si esta posee alguna celebridad. Su algoritmo deberá resolver el problema en tiempo lineal, su solución no podrá implicar un tiempo cuadrático (for o while anidados) o superior.

Listing 5: Programa llamado *ej5.pl*

```
#!/usr/bin/perl
use List::Util qw(max);
sub Celebrity {
  my ($mat) = @_;
  for(my $i = 0; $i < @$mat; $i++){
    $mat->[$i][$i] -= 1;
    my $ide = $mat->[$i][$i];
    my @row = @{$mat->[$i]};
    if($ide == 0 && max(@row) == 0){
      my $sum = 0;
      for(my $j = 0; $j < @$mat; $j++){
        $sum += $mat->[$j][$i];
      }
      if($sum == @$mat - 1){
        return 1;
      }
      return 0;
    }
  }
  return 0;
}
@mat = ([1,1,0], [1,1,0], [0,1,0], [1,1,0]);
$cel = Celebrity(\@mat);

print $cel."\n";
```

Nuevamente se utiliza el módulo *List::Util*, para tener acceso a la subrutina *max*. Se crea la subrutina *Celebrity*. En esta, se ingresa como parámetro una referencia a un arreglo bidimensional. Primero se resta 1 al elemento de la diagonal principal, para obtener 0 en

dicha posición, luego, si en la fila, el máximo elemento es 0, quiere decir que toda la fila son ceros, ello significa que estamos ante una posible celebridad, que es la única que puede haber en el arreglo bidimensional. Si no se encuentra una fila que contenga solamente ceros, la subrutina retorna 0 (false). Después, se itera en la columna de la celebridad, la cual, al sumarla, debe resultar la cantidad de filas menos 1, con ello nos aseguramos que estamos hablando de una celebridad, por lo tanto retornamos 1 (true), si esto no se cumple, la subrutina termina y retorna 0 (false).