Ejercicio 6

En el punto dos testearia cada función desde la más “primitiva” como por ejemplo:

```
sub esUnDecimalEntero{
   my($numEnDecimal)=shift @_;
   return ($numEnDecimal-int($numEnDecimal)>0);
}
```

hasta la más compleja como

Esto lo haría a medida que voy implementando cada función, por que al testearlas yo me garantizo que cuando las utilice dentro de otras funciones si surge algún problema estaría seguro que el error se encuentra en esa función que utilice .

```
sub equivalenciaHexaABinario {
   my ($numEnHexa) = shift @_;
   my %hexaABinario = (0 => "0000",
         1             => "0001",
         2             => "0010",
         3             => "0011",
         4             => "0100",
         5             => "0101",
         6             => "0110",
         7             => "0111",
         8             => "1000",
         9             => "1001",
         "A"           => "1010",
         "B"           => "1011",
         "C"           => "1100",
         "D"           => "1101",
         "E"           => "1110",
         "F"           => "1111");
   return $hexaABinario{$numEnHexa};
}
```

En esta función verificaria si al pasarle como parametro un número Hexadecimal me devuelve el correcto número Binario
verificaria si lo que devuelve es un string con 4 dígitos
por ejemplo
equivalenciaHexaABinario(“F”) debe retornar ”1111”

```
sub equivalenciaBinarioAHexa {
   my ($numEnHexa) = shift @_;
   my %binarioHexa = ("0000"=>"0",
         "0001"=>"1",
         "0010"=>"2",
         "0011"=>"3",
         "0100"=>"4",
         "0101"=>"5",
```

```
        "0110"=>"6",
        "0111"=>"7",
        "1000"=>"8",
        "1001"=>"9",
        "1010"=>"A",
        "1011"=>"B",
        "1100"=>"C",
        "1101"=>"D",
        "1110"=>"E",
        "1111"=>"F");
    return $binarioHexa{$numEnHexa};
}
```

En esta función verificaria si al pasarle como parametro un número Binario me devuelve el correcto número Hexadecimal
verificaría si lo que devuelve es un string con de un dígito.
equivalenciaBinarioAHexa("1111") debe retornar "F"

```
sub esUnDecimalEntero{
    my($numEnDecimal)=shift @_;
    return ($numEnDecimal-int($numEnDecimal)>0);
}
```

Pondría un caso con decimal fraccionario y esperaría que me retorne 0(false).
ejemplo:
esUnDecimalEntero(1,1) me retorne 0
Pondría otro caso con un decimal entero y esperaría que me retorne 1(true).
esUnDecimalEntero(1) me retorne 1

```
sub restoDeUnaDivision{
    my($dividendo)=shift @_;
    my($divisor)=shift @_;
    my$resultado=$dividendo/$divisor;
    my $resto=$dividendo-int($divisor*int($resultado));
    return ($resto);
}
```

En esta función verificaría si al pasarle el dividendo y el divisor me retorna el correcto resto.
ejemplo:
restoDeUnaDivision(9,3) me retorne 0
También testearia que si el divisor es cero retorne un error.
restoDeUnaDivision(9,0) me retorne "no se permite la división por cero")

```
sub darCantDeDigitosMult4{
   my ($cadena) = shift @_;
   my $cadenaAux="$num";
   my $cantDeCeros=(length($cadena)/4- int(length($cadena)/4))*4;
   for(1..$cantDeCeros){
       $cadenaAux='0'.$numAux    }
   return $cadenaAux ;
}
```

En esta función testearia que al ingresar como parámetro un número me retorne una cadena con una cantidad de dígitos que sea múltiplo de 4.
Testearia que si le doy un numero que no tenga una cantidad de digitos multiplo de 4 me devuelva una cadena con los primeros n ejemplo:
darCantDeDigitosMult4(11) me retorne “0011”
dígitos en cero para que la cadena total tenga una cantidad de digitos múltiplo de 4 y seguida por el string del número que se ingresó por parámetro.
darCantDeDigitosMult4(1111) me retorne “1111”

```
sub agruparDe{
   my ($cadena) = shift @_;
   my ($cantAgrupacion)=shift @_;
   my $cadenaAux = $cadena;
   my $agrupacion='';
   my $chopAux=0;
   for(1..$cantAgrupacion){
       $chopAux=chop($cadenaAux);
       $agrupacion= $chopAux. $agrupacion;
   }
   return($agrupacion);
}
```

En esta función testear que al pasarle un string me retorne un string de los ultimos n dígitos pasado por parametro.
Testearia que al pasarle como parámetro agrupar(unString,4)me retorne los últimos 4 dígitos del string.
ejemplo agruparDe(“1234567”,4)=”4567”

```
sub eliminarLosUltimosN{
   my($cadena)=shift @_;
   my($n)=shift @_;
   my$cadenaAux=$cadena;
   for(1..$n){
```

```
        chop($cadenaAux);
    }
    return $cadenaAux;
}
```

En esta función testear que al pasarle un string me retorne un string sin los ultimos n dígitos pasado por parametro.
Testearia que al pasarle como parámetro agrupar(un String,4)me retorne un string sin los últimos 4 dígitos del mismo.
ejemplo agruparDe(“1234567”,4)=”123”

```
sub convertirBinarioADecimal{
    my($nAConvetir)=shift @_;
    my$numeroAux=$nAConvetir;
    my$cantDigitos=length($nAConvetir);
    my$numeroConvertido=0;
    my $exponenteAux=0;
    for my $index(0..$cantDigitos-1){
        $exponenteAux=2**$index;

$numeroConvertido=$numeroConvertido+(chop($numeroAux)*$exponenteAu
x);
    }
    return("$numeroConvertido")
}
```

En esta función testear que al pasarle una cadena que represente un número en Binario me retorne la cadena con el numero convertido en Decimal.
Ejemplo

convertirBinarioADecimal(“10”)=”2”

```
sub convertirDecimalABinario{
    my($numEnBinario)=shift @_;
    my$numAConvertir=$numEnBinario;
```

```
    my @listaNumConvertido=();
    my $ultimoTerminoProcesado=0;
    while($numAConvertir>=2){

push(@listaNumConvertido,restoDeUnaDivision($numAConvertir,2));
        $ultimoTerminoProcesado=$numAConvertir;
        $numAConvertir=$numAConvertir/2;
        $numAConvertir =int($numAConvertir);
    }
    push(@listaNumConvertido,int($ultimoTerminoProcesado/2));
    return(reverse(@listaNumConvertido))
}
```

En esta función testear que al pasarle una cadena que represente un numero en decimal me retorne la cadena con el numero convertido en Binario.
Ejemplo

convertirDecimalABinario(“2”) me retorne ”10”

```
sub convertirHexaABinario {
    my ($numeroEnHexa) = shift @_;
    my $numeroAuxiliar=$numeroEnHexa;
    my $resultado = "";
    for (1 .. length($numeroEnHexa)) {
        $resultado=
equivalenciaHexaABinario(chop($numeroAuxiliar)).$resultado;
    }
    return ($resultado);
}
```

En esta función testear que al pasarle una cadena que represente un número en hexadecimal me retorna la cadena con el numero convertido en Binario.
Ejemplo

convertirBinarioADecimal(“FF”) me retorne ”11111111”

```
sub convertirDecimalAHexa {
    my ($numeroEnDecimal) = shift @_;
    my $numeroAuxiliar=$numeroEnDecimal;
    my $numeroEnBinario=convertirDecimalABinario($numeroAuxiliar);
    return
convertirBinarioAHexa(cantDeDigitosMult4($numeroEnBinario));
}
```

En esta función testear que al pasarle una cadena que represente un numero en decimal me retorne la cadena con el numero convertido en Hexadecimal.
Ejemplo

convertirDecimalAHexa(“16”) me retorne ”10”

```
sub convertirBinarioAHexa{
   my ($numeroEnBinario) = shift @_;
   my $resultado = "";
   my $numeroAuxiliar=cantDeDigitosMult4("$numeroEnBinario");
   my $agrupacion='';
   for my$index(1.. (length($numeroAuxiliar)/4)){
       $agrupacion=agruparDe($numeroAuxiliar,4);
       $numeroAuxiliar=eliminarLosUltimosN($numeroAuxiliar,4);
       $resultado=
equivalenciaBinarioAHexa($agrupacion).$resultado;
   }
   return $resultado;
}
```

En esta función testear que al pasarle una cadena que represente un número en Binario me retorne la cadena con el número convertido en Hexadecimal.
Ejemplo

convertirBinarioAHexa(“1010”) me retorne ”A”

Otro caso que testear es que al pasarle una cadena que represente el número en Binario que no tenga una cantidad de dígitos múltiplos de 4 me retorna la cadena que represente el número convertido en hexadecimal con los n ceros necesarios como prefijo para que al sumar la cantidad de dígitos de la cadena total este valor sea múltiplo de cuatro
convertirBinarioAHexa(“10”) me retorne ”2”

```
sub convertir HexADecimal{
   my ($numeroEnHexa) = shift @_;
   return
convertirBinarioADecimal(convertirHexaABinario($numeroEnHexa));
}
```

En esta función testearé que al pasarle una cadena que represente un número en Hexadecimal me retorne la cadena con el número convertido en Decimal .

Ejemplo

convertirHexaADecimal(“AA”) me retorne ”10101010”

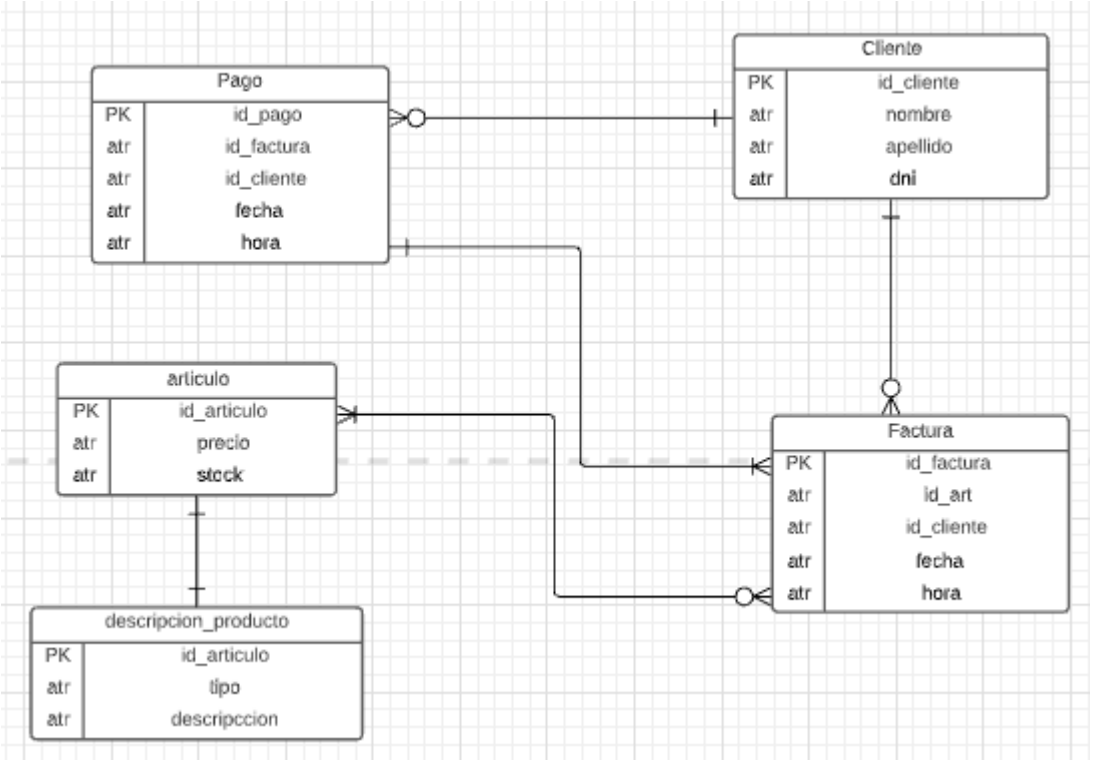
```
sub convertir {
    my $base1=shift @_;
    my $numero=uc(shift @_);
    my $base2=shift @_;
    my $conversionRealizada='';
    if($base1==2and$base2==10){
        $conversionRealizada= convertirBinarioADecimal($numero)#ok
    }
    elsif($base1==2and$base2==16){
        $conversionRealizada=convertirBinarioAHexa($numero)#ok
    }
    elsif($base1==16and$base2==10){
        $conversionRealizada=convertirHexaADecimal($numero)
    }
    elsif($base1==16and$base2==2){
        $conversionRealizada=convertirHexaABinario($numero)#ok
    }
    elsif($base1==10and$base2==2){
        $conversionRealizada=convertirDecimalABinario($numero)#ok
    }
    elsif($base1==10and$base2==16){
        $conversionRealizada=convertirDecimalAHexa($numero)
    }
    else{
        print("Base seleccionada invalida")
    }

    return $conversionRealizada;
}
```

En esta función verifica si la conversion en cada caso combinando las bases 2,10,16 y una cadena que representa un número en la base 1 me retorne el valor verdadero.
verificaria si pongo una base invalida me devuelva un msj indicando el error

También verificará que la cadena puede ser ingresada en mayúscula o en minuscula
convertir(16,“a”,10) debe retornar ”10”

Ejercicio 7



ejercicio 8

select sum(precio)
where id_cliente=3
from left join Factura on Pago inner join Articulo inner join cliente on Factura.id_factura=Pago.id_factura on Articulo.id_articulo=Factura.id_articulo

select SUM(precio) where id_cliente= “1” from Factura inner join articulo on Factura.id_articulo=articulo.id_articulo left join Pago on Factura.id_factura=Pago.id_factura

ejercicio 9

select id_cliente,nombre, apellido,dni
from Cliente inner join Factura on Cliente.id_cliente= Factura.id_cliente left join Pago on Pago.id_factura=Factura.id_factura