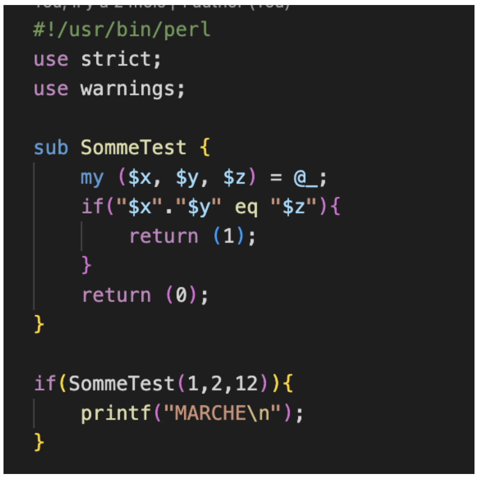
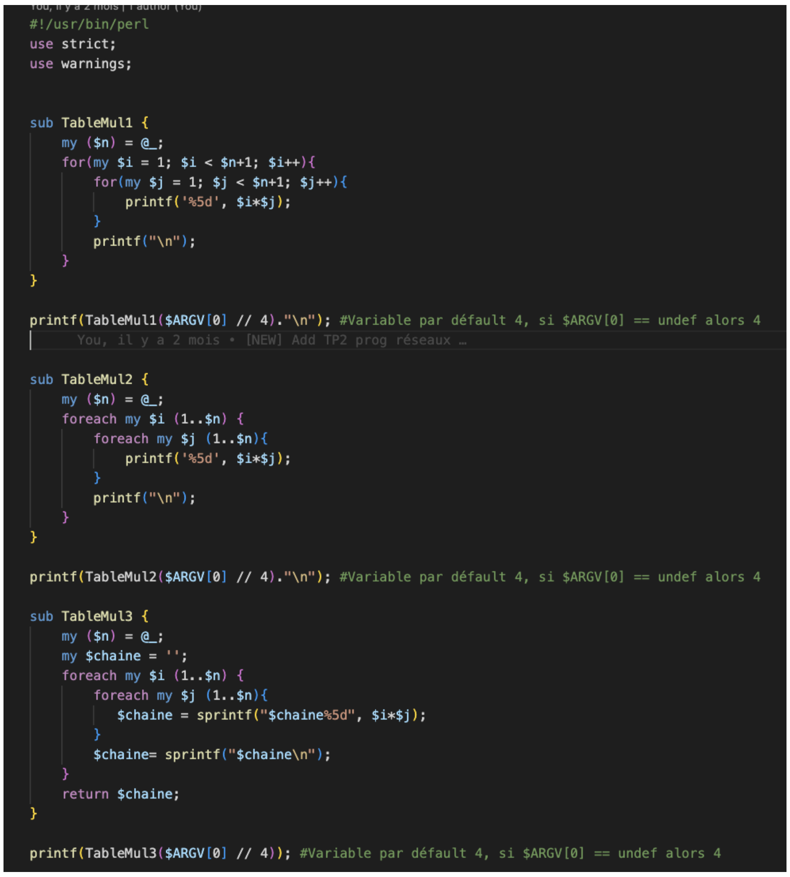
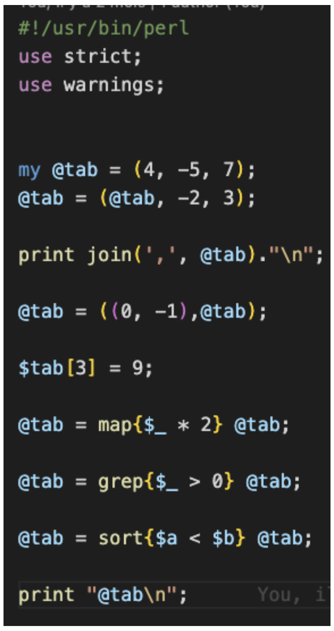
TP1



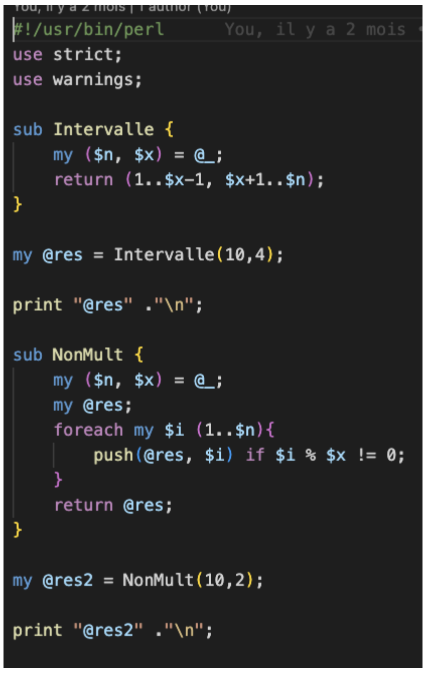
```perl
#!/usr/bin/perl
use strict;
use warnings;

sub SommeTest {
    my ($x, $y, $z) = @_;
    if("$x"."$y" eq "$z"){
        return (1);
    }
    return (0);
}

if(SommeTest(1,2,12)){
    printf("MARCHE\n");
}
```



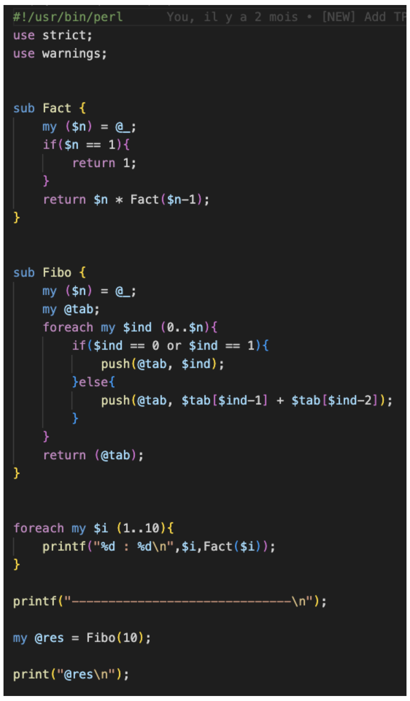
```perl
#!/usr/bin/perl
use strict;
use warnings;


sub TableMul1 {
    my ($n) = @_;
    for(my $i = 1; $i < $n+1; $i++){
        for(my $j = 1; $j < $n+1; $j++){
            printf('%5d', $i*$j);
        }
        printf("\n");
    }
}

printf(TableMul1($ARGV[0] // 4)."\n"); #Variable par défault 4, si $ARGV[0] == undef alors 4
      You, il y a 2 mois • [NEW] Add TP2 prog réseaux …

sub TableMul2 {
    my ($n) = @_;
    foreach my $i (1..$n) {
        foreach my $j (1..$n){
            printf('%5d', $i*$j);
        }
        printf("\n");
    }
}

printf(TableMul2($ARGV[0] // 4)."\n"); #Variable par défault 4, si $ARGV[0] == undef alors 4

sub TableMul3 {
    my ($n) = @_;
    my $chaine = '';
    foreach my $i (1..$n) {
        foreach my $j (1..$n){
           $chaine = sprintf("$chaine%5d", $i*$j);
        }
        $chaine= sprintf("$chaine\n");
    }
    return $chaine;
}

printf(TableMul3($ARGV[0] // 4)); #Variable par défault 4, si $ARGV[0] == undef alors 4
```



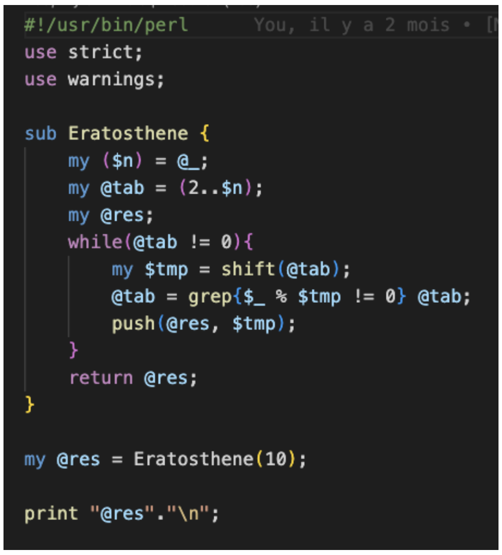
```perl
#!/usr/bin/perl
use strict;
use warnings;


my @tab = (4, -5, 7);
@tab = (@tab, -2, 3);

print join(',', @tab)."\n";

@tab = ((0, -1),@tab);

$tab[3] = 9;

@tab = map{$_ * 2} @tab;

@tab = grep{$_ > 0} @tab;

@tab = sort{$a < $b} @tab;

print "@tab\n";         You, i
```



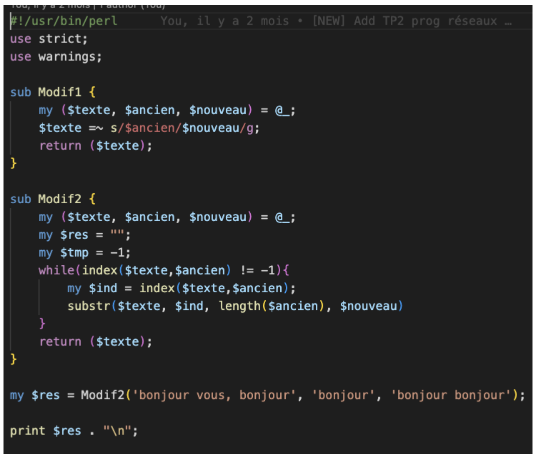
```perl
#!/usr/bin/perl        You, il y a 2 mois •
use strict;
use warnings;

sub Intervalle {
    my ($n, $x) = @_;
    return (1..$x-1, $x+1..$n);
}

my @res = Intervalle(10,4);

print "@res" ."\n";

sub NonMult {
    my ($n, $x) = @_;
    my @res;
    foreach my $i (1..$n){
        push(@res, $i) if $i % $x != 0;
    }
    return @res;
}

my @res2 = NonMult(10,2);

print "@res2" ."\n";
```



```perl
#!/usr/bin/perl      You, il y a 2 mois • [NEW] Add TP
use strict;
use warnings;


sub Fact {
    my ($n) = @_;
    if($n == 1){
        return 1;
    }
    return $n * Fact($n-1);
}


sub Fibo {
    my ($n) = @_;
    my @tab;
    foreach my $ind (0..$n){
        if($ind == 0 or $ind == 1){
            push(@tab, $ind);
        }else{
            push(@tab, $tab[$ind-1] + $tab[$ind-2]);
        }
    }
    return (@tab);
}


foreach my $i (1..10){
    printf("%d : %d\n",$i,Fact($i));
}

printf("------------------------------\n");

my @res = Fibo(10);

print("@res\n");
```

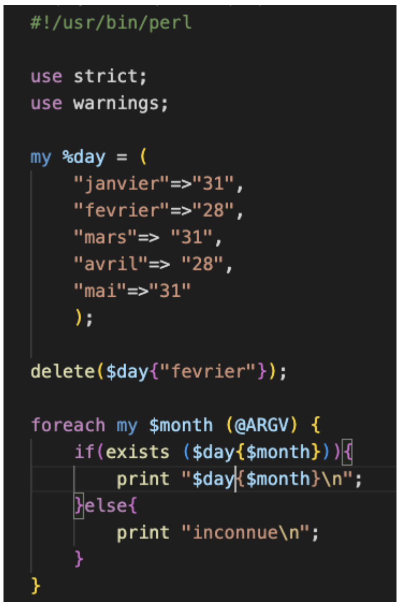


```perl
#!/usr/bin/perl        You, il y a 2 mois • [
use strict;
use warnings;

sub Eratosthene {
    my ($n) = @_;
    my @tab = (2..$n);
    my @res;
    while(@tab != 0){
        my $tmp = shift(@tab);
        @tab = grep{$_ % $tmp != 0} @tab;
        push(@res, $tmp);
    }
    return @res;
}

my @res = Eratosthene(10);

print "@res"."\n";
```

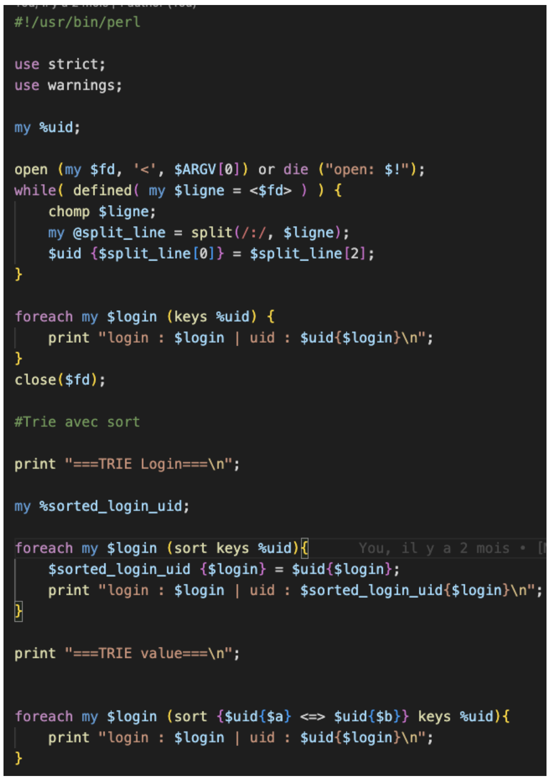


```perl
#!/usr/bin/perl       You, il y a 2 mois • [NEW] Add TP2 prog réseaux …
use strict;
use warnings;

sub Modif1 {
    my ($texte, $ancien, $nouveau) = @_;
    $texte =~ s/$ancien/$nouveau/g;
    return ($texte);
}

sub Modif2 {
    my ($texte, $ancien, $nouveau) = @_;
    my $res = "";
    my $tmp = -1;
    while(index($texte,$ancien) != -1){
        my $ind = index($texte,$ancien);
        substr($texte, $ind, length($ancien), $nouveau)
    }
    return ($texte);
}

my $res = Modif2('bonjour vous, bonjour', 'bonjour', 'bonjour bonjour');

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

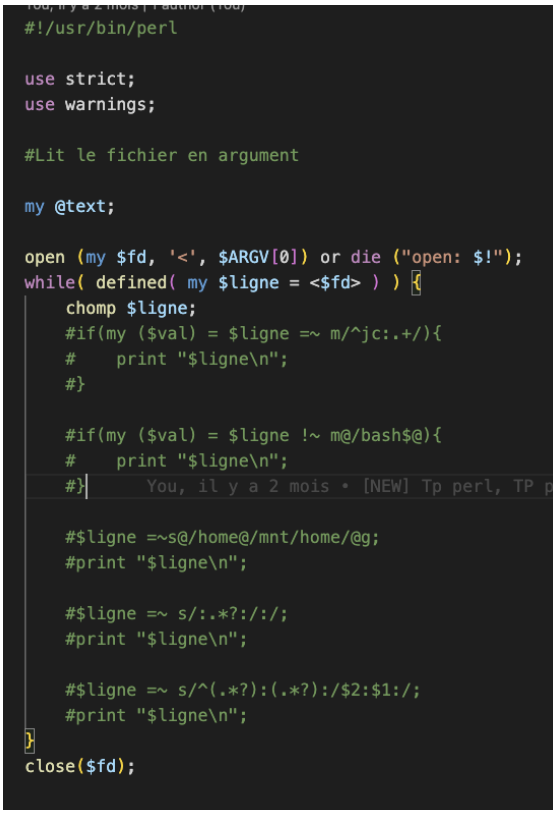
TP2



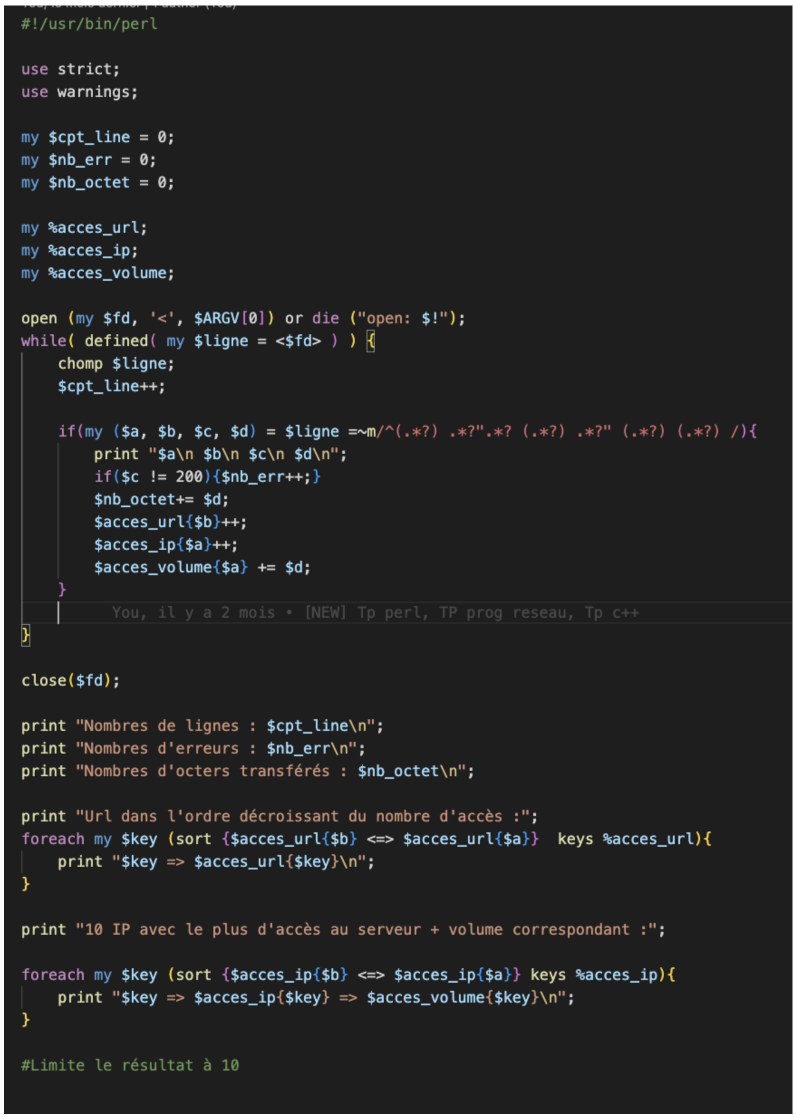
```perl
#!/usr/bin/perl

use strict;
use warnings;

my %day = (
    "janvier"=>"31",
    "fevrier"=>"28",
    "mars"=> "31",
    "avril"=> "28",
    "mai"=>"31"
    );

delete($day{"fevrier"});

foreach my $month (@ARGV) {
    if(exists ($day{$month})){
        print "$day{$month}\n";
    }else{
        print "inconnue\n";
    }
}
```



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

use strict;
use warnings;

my %uid;

open (my $fd, '<', $ARGV[0]) or die ("open: $!");
while( defined( my $ligne = <$fd> ) ) {
    chomp $ligne;
    my @split_line = split(/:/, $ligne);
    $uid {$split_line[0]} = $split_line[2];
}

foreach my $login (keys %uid) {
    print "login : $login | uid : $uid{$login}\n";
}
close($fd);

#Trie avec sort

print "===TRIE Login===\n";

my %sorted_login_uid;

foreach my $login (sort keys %uid){        You, il y a 2 mois • [
    $sorted_login_uid {$login} = $uid{$login};
    print "login : $login | uid : $sorted_login_uid{$login}\n";
}

print "===TRIE value===\n";


foreach my $login (sort {$uid{$a} <=> $uid{$b}} keys %uid){
    print "login : $login | uid : $uid{$login}\n";
}
```

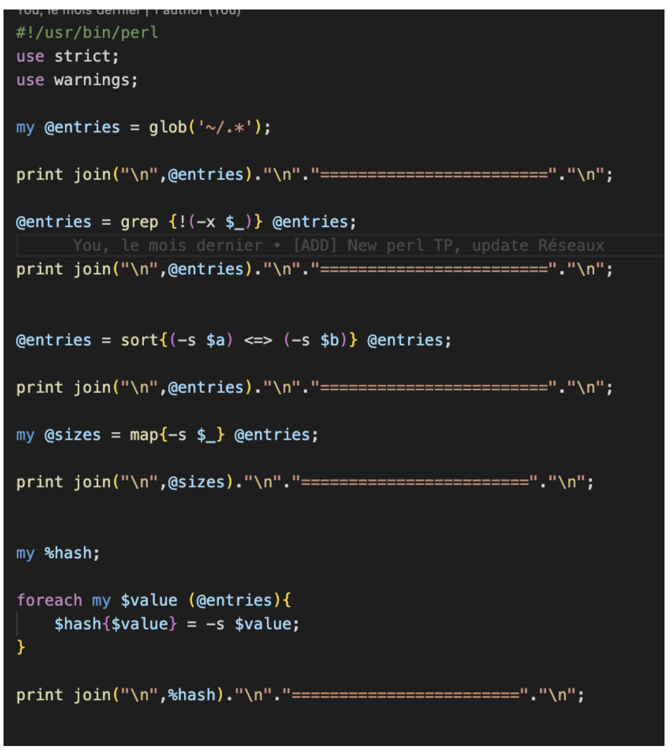


```perl
#!/usr/bin/perl

use strict;
use warnings;

#Lit le fichier en argument

my @text;

open (my $fd, '<', $ARGV[0]) or die ("open: $!");
while( defined( my $ligne = <$fd> ) ) {
    chomp $ligne;
    #if(my ($val) = $ligne =~ m/^jc:.+/){
    #    print "$ligne\n";
    #}

    #if(my ($val) = $ligne !~ m@/bash$@){
    #    print "$ligne\n";
    #}        You, il y a 2 mois • [NEW] Tp perl, TP p

    #$ligne =~s@/home@/mnt/home/@g;
    #print "$ligne\n";

    #$ligne =~ s/:.*?:/:/;
    #print "$ligne\n";

    #$ligne =~ s/^(.*?):(.*?):/$2:$1:/;
    #print "$ligne\n";
}
close($fd);
```

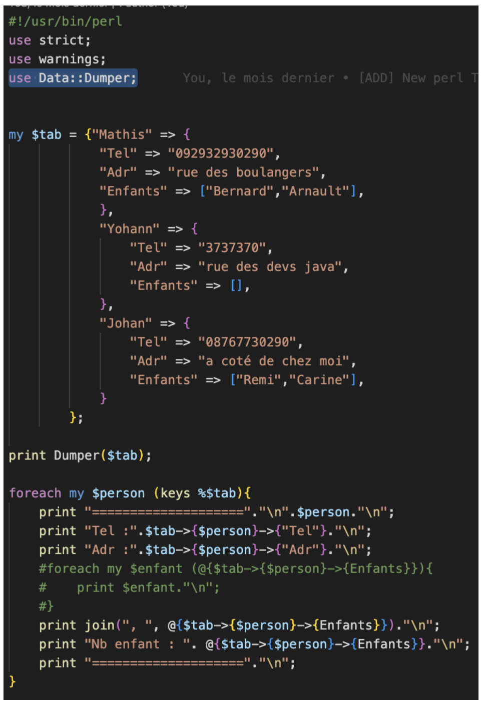


```perl
#!/usr/bin/perl

use strict;
use warnings;

my $cpt_line = 0;
my $nb_err = 0;
my $nb_octet = 0;

my %acces_url;
my %acces_ip;
my %acces_volume;

open (my $fd, '<', $ARGV[0]) or die ("open: $!");
while( defined( my $ligne = <$fd> ) ) {
    chomp $ligne;
    $cpt_line++;

    if(my ($a, $b, $c, $d) = $ligne =~m/^(.*?) .*?".*? (.*?) .*?" (.*?) (.*?) /){
        print "$a\n $b\n $c\n $d\n";
        if($c != 200){$nb_err++;}
        $nb_octet+= $d;
        $acces_url{$b}++;
        $acces_ip{$a}++;
        $acces_volume{$a} += $d;
    }
            You, il y a 2 mois • [NEW] Tp perl, TP prog reseau, Tp c++
}

close($fd);

print "Nombres de lignes : $cpt_line\n";
print "Nombres d'erreurs : $nb_err\n";
print "Nombres d'octers transférés : $nb_octet\n";

print "Url dans l'ordre décroissant du nombre d'accès :";
foreach my $key (sort {$acces_url{$b} <=> $acces_url{$a}}  keys %acces_url){
    print "$key => $acces_url{$key}\n";
}

print "10 IP avec le plus d'accès au serveur + volume correspondant :";

foreach my $key (sort {$acces_ip{$b} <=> $acces_ip{$a}} keys %acces_ip){
    print "$key => $acces_ip{$key} => $acces_volume{$key}\n";
}

#Limite le résultat à 10
```

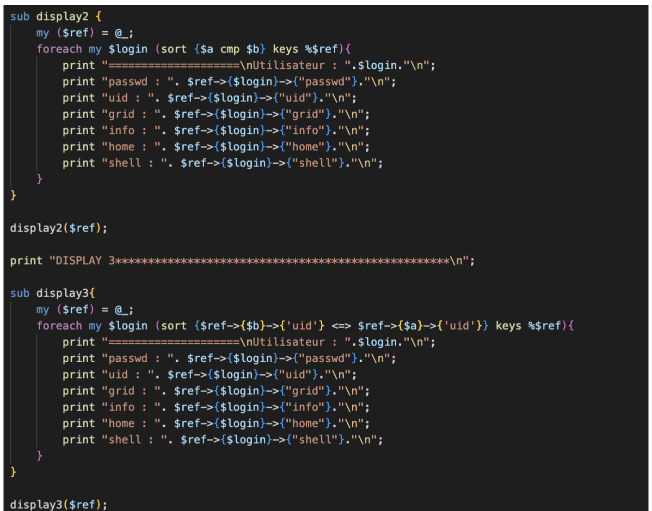
TP3



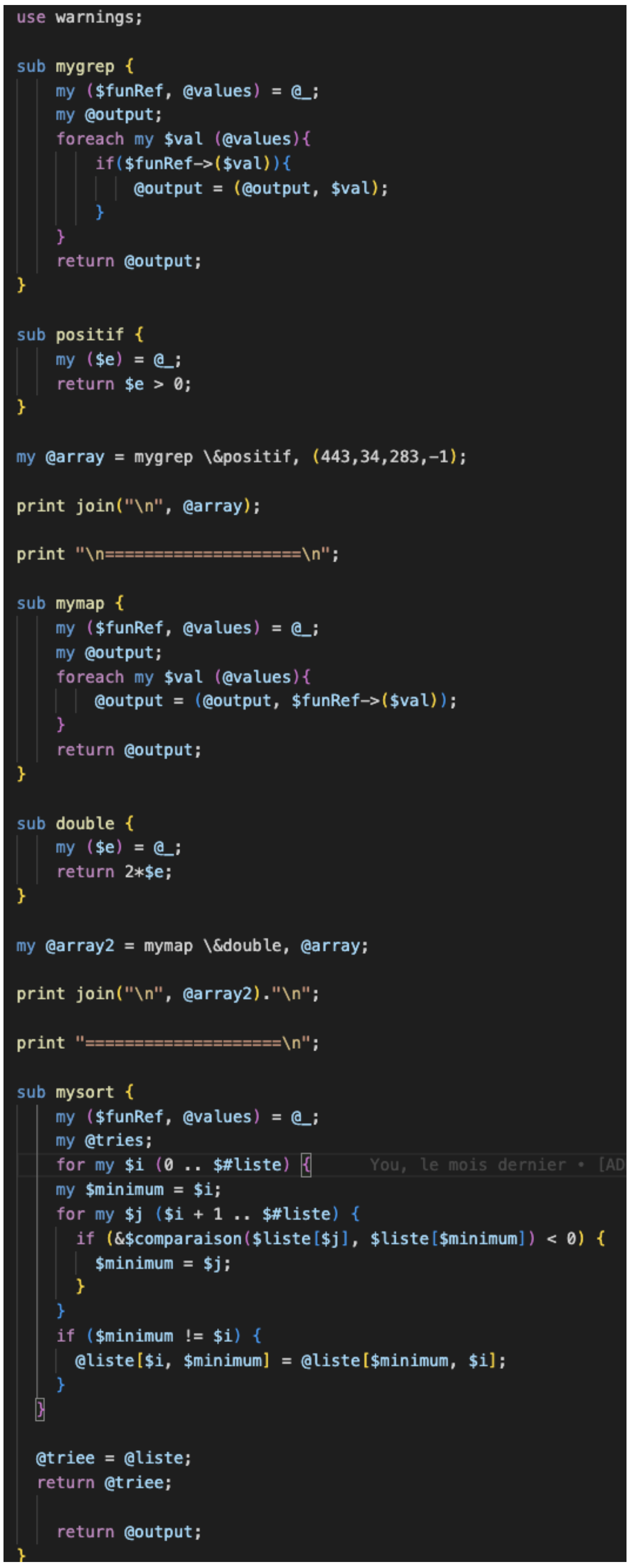
```perl
You, le mois dernier | 1 author (You)
#!/usr/bin/perl
use strict;
use warnings;

my @entries = glob('~/.*');

print join("\n",@entries)."\n"."========================="."\n";

@entries = grep {!(-x $_)} @entries;
        You, le mois dernier • [ADD] New perl TP, update Réseaux
print join("\n",@entries)."\n"."========================="."\n";


@entries = sort{(-s $a) <=> (-s $b)} @entries;

print join("\n",@entries)."\n"."========================="."\n";

my @sizes = map{-s $_} @entries;

print join("\n",@sizes)."\n"."========================="."\n";


my %hash;

foreach my $value (@entries){
    $hash{$value} = -s $value;
}

print join("\n",%hash)."\n"."========================="."\n";
```



```perl
#!/usr/bin/perl
use strict;
use warnings;
use Data::Dumper;       You, le mois dernier • [ADD] New perl T


my $tab = {"Mathis" => {
            "Tel" => "092932930290",
            "Adr" => "rue des boulangers",
            "Enfants" => ["Bernard","Arnault"],
            },
            "Yohann" => {
                "Tel" => "3737370",
                "Adr" => "rue des devs java",
                "Enfants" => [],
            },
            "Johan" => {
                "Tel" => "08767730290",
                "Adr" => "a coté de chez moi",
                "Enfants" => ["Remi","Carine"],
            }
        };

print Dumper($tab);

foreach my $person (keys %$tab){
    print "===================="."\n".$person."\n";
    print "Tel :".$tab->{$person}->{"Tel"}."\n";
    print "Adr :".$tab->{$person}->{"Adr"}."\n";
    #foreach my $enfant (@{$tab->{$person}->{Enfants}}){
    #    print $enfant."\n";
    #}
    print join(", ", @{$tab->{$person}->{Enfants}})."\n";
    print "Nb enfant : ". @{$tab->{$person}->{Enfants}}."\n";
    print "===================="."\n";
}
```

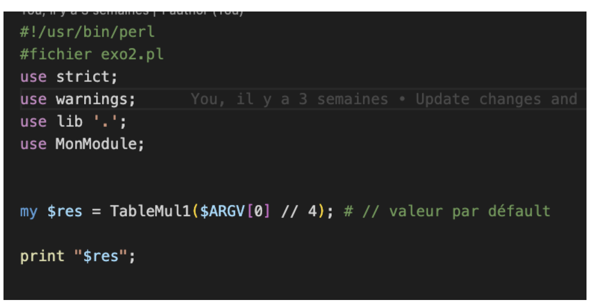


```perl
sub display2 {
    my ($ref) = @_;
    foreach my $login (sort {$a cmp $b} keys %$ref){
        print "====================\nUtilisateur : ".$login."\n";
        print "passwd : ". $ref->{$login}->{"passwd"}."\n";
        print "uid : ". $ref->{$login}->{"uid"}."\n";
        print "grid : ". $ref->{$login}->{"grid"}."\n";
        print "info : ". $ref->{$login}->{"info"}."\n";
        print "home : ". $ref->{$login}->{"home"}."\n";
        print "shell : ". $ref->{$login}->{"shell"}."\n";
    }
}

display2($ref);

print "DISPLAY 3*****************************************************\n";

sub display3{
    my ($ref) = @_;
    foreach my $login (sort {$ref->{$b}->{'uid'} <=> $ref->{$a}->{'uid'}} keys %$ref){
        print "====================\nUtilisateur : ".$login."\n";
        print "passwd : ". $ref->{$login}->{"passwd"}."\n";
        print "uid : ". $ref->{$login}->{"uid"}."\n";
        print "grid : ". $ref->{$login}->{"grid"}."\n";
        print "info : ". $ref->{$login}->{"info"}."\n";
        print "home : ". $ref->{$login}->{"home"}."\n";
        print "shell : ". $ref->{$login}->{"shell"}."\n";
    }
}

display3($ref);
```

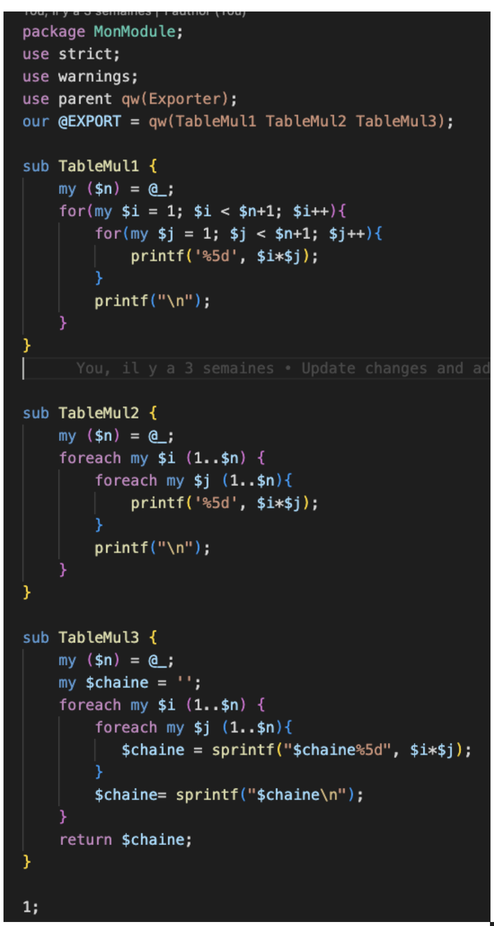


```perl
use warnings;

sub mygrep {
    my ($funRef, @values) = @_;
    my @output;
    foreach my $val (@values){
        if($funRef->($val)){
            @output = (@output, $val);
        }
    }
    return @output;
}

sub positif {
    my ($e) = @_;
    return $e > 0;
}

my @array = mygrep \&positif, (443,34,283,-1);

print join("\n", @array);

print "\n====================\n";

sub mymap {
    my ($funRef, @values) = @_;
    my @output;
    foreach my $val (@values){
        @output = (@output, $funRef->($val));
    }
    return @output;
}

sub double {
    my ($e) = @_;
    return 2*$e;
}

my @array2 = mymap \&double, @array;

print join("\n", @array2)."\n";

print "====================\n";

sub mysort {
    my ($funRef, @values) = @_;
    my @tries;
    for my $i (0 .. $#liste) {          You, le mois dernier • [AD
    my $minimum = $i;
    for my $j ($i + 1 .. $#liste) {
      if (&$comparaison($liste[$j], $liste[$minimum]) < 0) {
        $minimum = $j;
      }
    }
    if ($minimum != $i) {
      @liste[$i, $minimum] = @liste[$minimum, $i];
    }
  }

  @triee = @liste;
  return @triee;

    return @output;
}
```

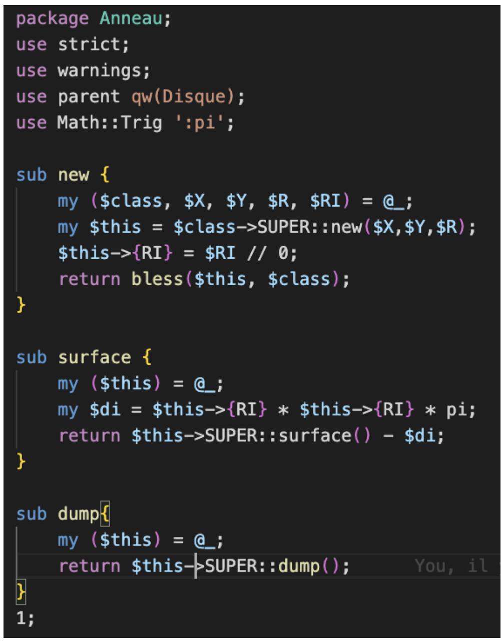
TP4



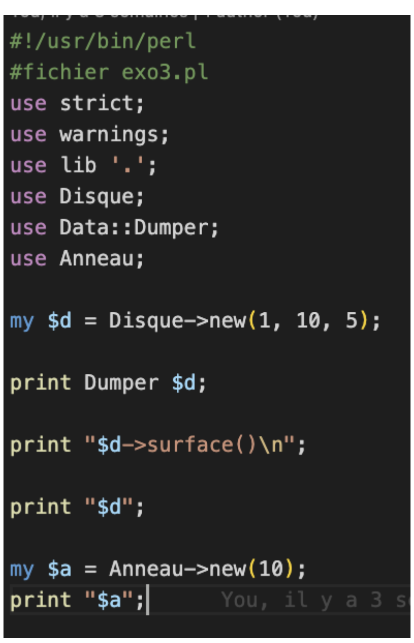
```perl
#!/usr/bin/perl
#fichier exo2.pl
use strict;
use warnings;         You, il y a 3 semaines • Update changes and
use lib '.';
use MonModule;


my $res = TableMul1($ARGV[0] // 4); # // valeur par défault

print "$res";
```



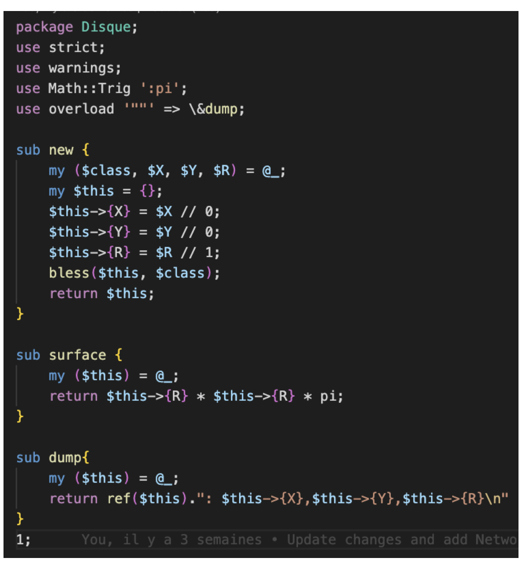
```perl
package MonModule;
use strict;
use warnings;
use parent qw(Exporter);
our @EXPORT = qw(TableMul1 TableMul2 TableMul3);

sub TableMul1 {
    my ($n) = @_;
    for(my $i = 1; $i < $n+1; $i++){
        for(my $j = 1; $j < $n+1; $j++){
            printf('%5d', $i*$j);
        }
        printf("\n");
    }
}
|     You, il y a 3 semaines • Update changes and ad

sub TableMul2 {
    my ($n) = @_;
    foreach my $i (1..$n) {
        foreach my $j (1..$n){
            printf('%5d', $i*$j);
        }
        printf("\n");
    }
}

sub TableMul3 {
    my ($n) = @_;
    my $chaine = '';
    foreach my $i (1..$n) {
        foreach my $j (1..$n){
          $chaine = sprintf("$chaine%5d", $i*$j);
        }
        $chaine= sprintf("$chaine\n");
    }
    return $chaine;
}

1;
```



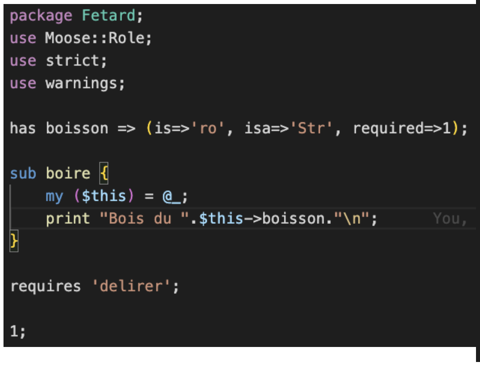
```perl
package Anneau;
use strict;
use warnings;
use parent qw(Disque);
use Math::Trig ':pi';

sub new {
    my ($class, $X, $Y, $R, $RI) = @_;
    my $this = $class->SUPER::new($X,$Y,$R);
    $this->{RI} = $RI // 0;
    return bless($this, $class);
}

sub surface {
    my ($this) = @_;
    my $di = $this->{RI} * $this->{RI} * pi;
    return $this->SUPER::surface() - $di;
}

sub dump{
    my ($this) = @_;
    return $this->SUPER::dump();       You, il
}
1;
```



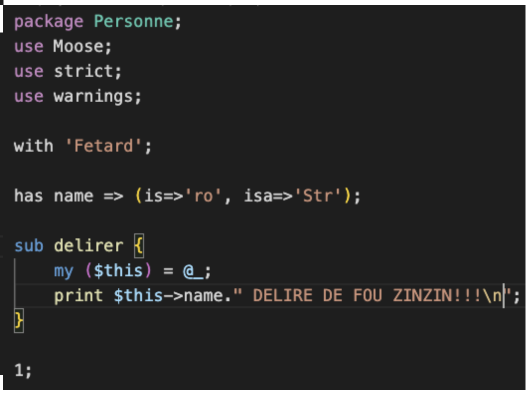
```perl
#!/usr/bin/perl
#fichier exo3.pl
use strict;
use warnings;
use lib '.';
use Disque;
use Data::Dumper;
use Anneau;

my $d = Disque->new(1, 10, 5);

print Dumper $d;

print "$d->surface()\n";

print "$d";

my $a = Anneau->new(10);
print "$a";       You, il y a 3 s
```



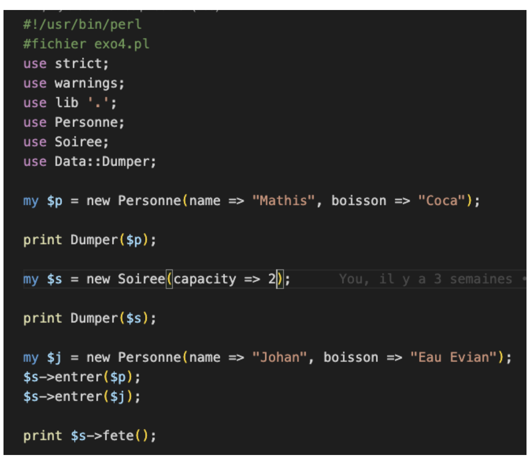
```perl
package Disque;
use strict;
use warnings;
use Math::Trig ':pi';
use overload '""' => \&dump;

sub new {
    my ($class, $X, $Y, $R) = @_;
    my $this = {};
    $this->{X} = $X // 0;
    $this->{Y} = $Y // 0;
    $this->{R} = $R // 1;
    bless($this, $class);
    return $this;
}

sub surface {
    my ($this) = @_;
    return $this->{R} * $this->{R} * pi;
}

sub dump{
    my ($this) = @_;
    return ref($this).": $this->{X},$this->{Y},$this->{R}\n"
}
1;       You, il y a 3 semaines • Update changes and add Netwo
```



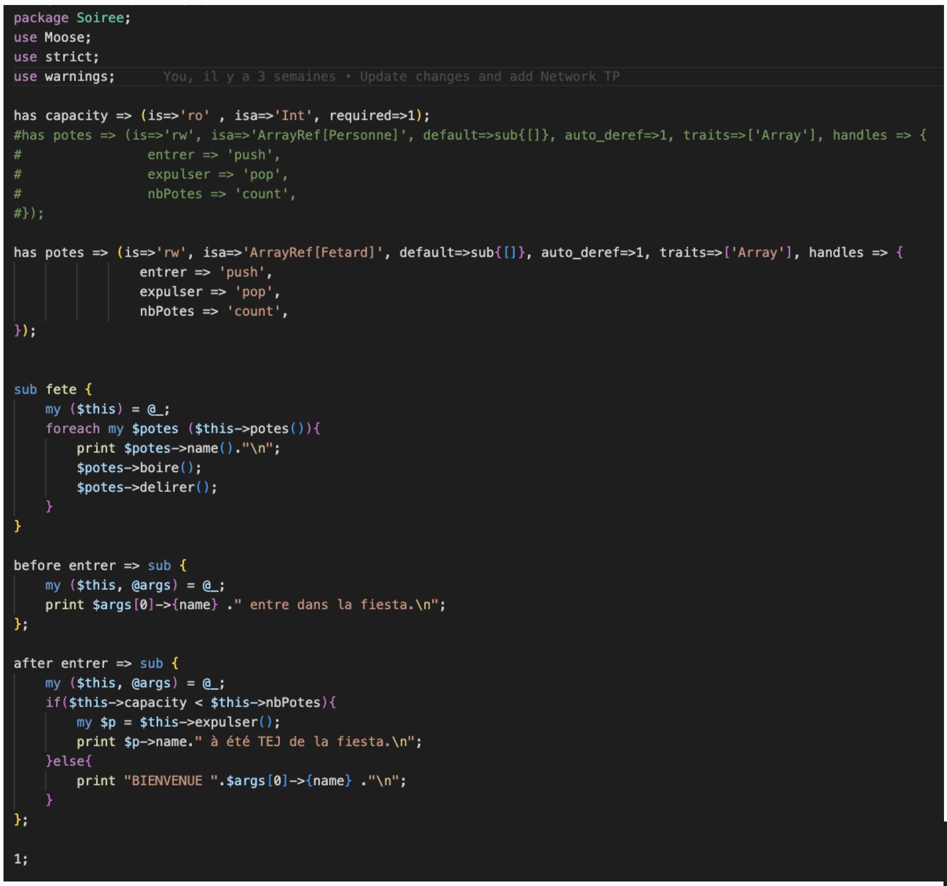
```perl
package Fetard;
use Moose::Role;
use strict;
use warnings;

has boisson => (is=>'ro', isa=>'Str', required=>1);

sub boire {
    my ($this) = @_;
    print "Bois du ".$this->boisson."\n";       You,
}

requires 'delirer';

1;
```



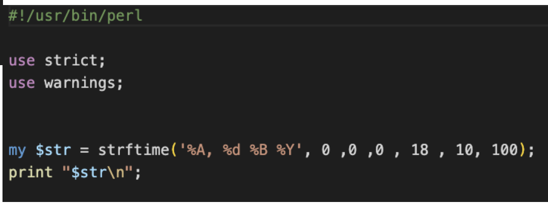
```perl
package Personne;
use Moose;
use strict;
use warnings;

with 'Fetard';

has name => (is=>'ro', isa=>'Str');

sub delirer {
    my ($this) = @_;
    print $this->name." DELIRE DE FOU ZINZIN!!!\n";
}

1;
```



```perl
#!/usr/bin/perl
#fichier exo4.pl
use strict;
use warnings;
use lib '.';
use Personne;
use Soiree;
use Data::Dumper;

my $p = new Personne(name => "Mathis", boisson => "Coca");

print Dumper($p);

my $s = new Soiree(capacity => 2);       You, il y a 3 semaines •

print Dumper($s);

my $j = new Personne(name => "Johan", boisson => "Eau Evian");
$s->entrer($p);
$s->entrer($j);

print $s->fete();
```

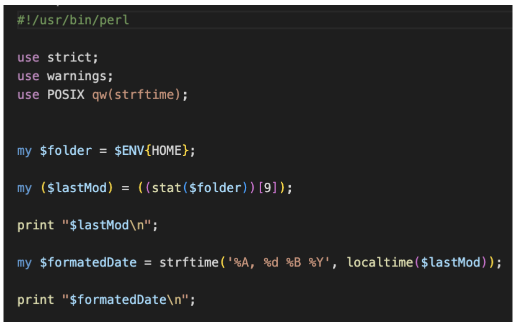


```perl
package Soiree;
use Moose;
use strict;
use warnings;      You, il y a 3 semaines • Update changes and add Network TP

has capacity => (is=>'ro' , isa=>'Int', required=>1);
#has potes => (is=>'rw', isa=>'ArrayRef[Personne]', default=>sub{[]}, auto_deref=>1, traits=>['Array'], handles => {
#                 entrer => 'push',
#                 expulser => 'pop',
#                 nbPotes => 'count',
#});

has potes => (is=>'rw', isa=>'ArrayRef[Fetard]', default=>sub{[]}, auto_deref=>1, traits=>['Array'], handles => {
                entrer => 'push',
                expulser => 'pop',
                nbPotes => 'count',
});


sub fete {
    my ($this) = @_;
    foreach my $potes ($this->potes()){
        print $potes->name()."\n";
        $potes->boire();
        $potes->delirer();
    }
}

before entrer => sub {
    my ($this, @args) = @_;
    print $args[0]->{name} ." entre dans la fiesta.\n";
};

after entrer => sub {
    my ($this, @args) = @_;
    if($this->capacity < $this->nbPotes){
        my $p = $this->expulser();
        print $p->name." à été TEJ de la fiesta.\n";
    }else{
        print "BIENVENUE ".$args[0]->{name} ."\n";
    }
};

1;
```

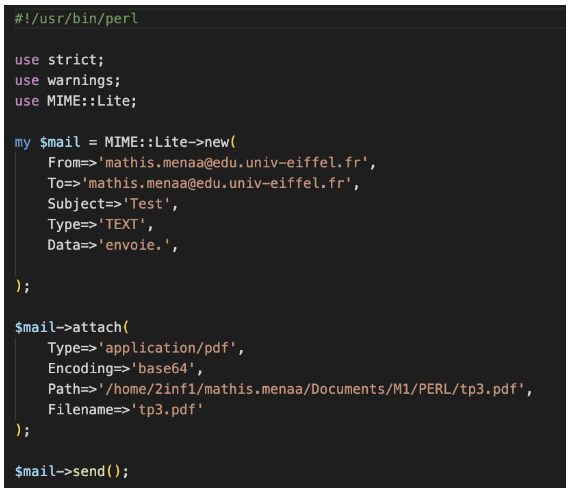


```perl
#!/usr/bin/perl

use strict;
use warnings;


my $str = strftime('%A, %d %B %Y', 0 ,0 ,0 , 18 , 10, 100);
print "$str\n";
```

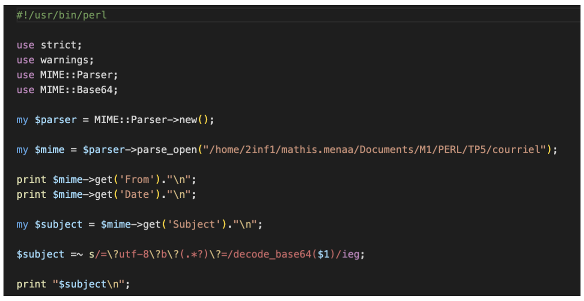
TP5



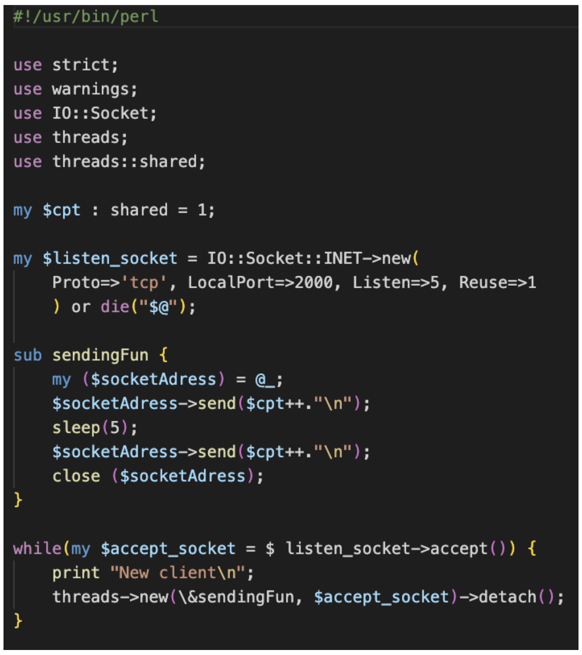
```perl
#!/usr/bin/perl

use strict;
use warnings;
use POSIX qw(strftime);


my $folder = $ENV{HOME};

my ($lastMod) = ((stat($folder))[9]);

print "$lastMod\n";

my $formatedDate = strftime('%A, %d %B %Y', localtime($lastMod));

print "$formatedDate\n";
```



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

use strict;
use warnings;
use MIME::Lite;

my $mail = MIME::Lite->new(
    From=>'mathis.menaa@edu.univ-eiffel.fr',
    To=>'mathis.menaa@edu.univ-eiffel.fr',
    Subject=>'Test',
    Type=>'TEXT',
    Data=>'envoie.',

);

$mail->attach(
    Type=>'application/pdf',
    Encoding=>'base64',
    Path=>'/home/2inf1/mathis.menaa/Documents/M1/PERL/tp3.pdf',
    Filename=>'tp3.pdf'
);

$mail->send();
```

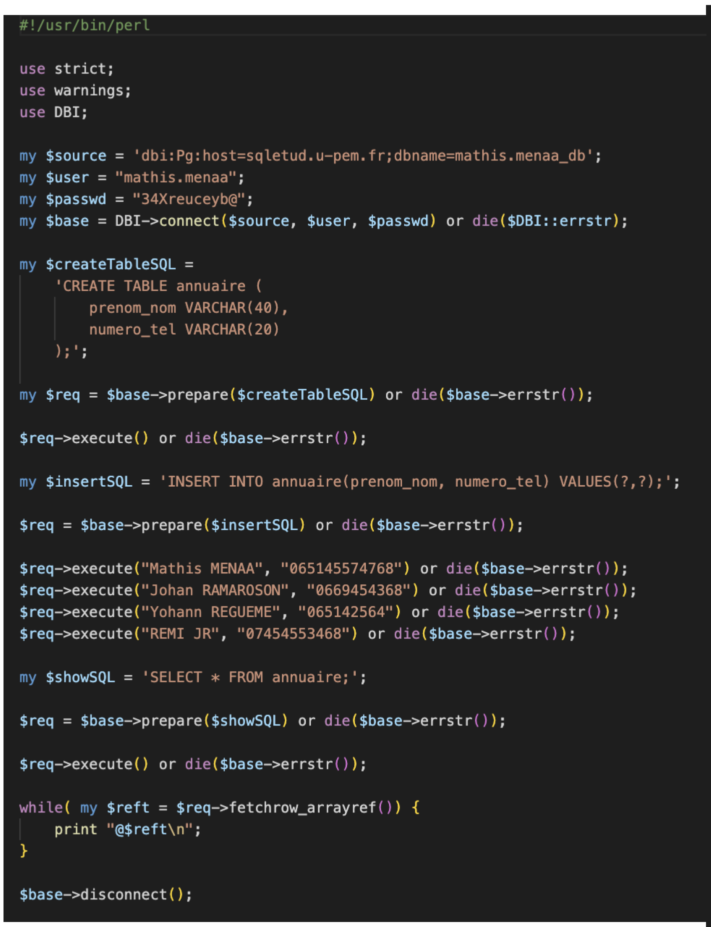


```perl
#!/usr/bin/perl

use strict;
use warnings;
use MIME::Parser;
use MIME::Base64;

my $parser = MIME::Parser->new();

my $mime = $parser->parse_open("/home/2inf1/mathis.menaa/Documents/M1/PERL/TP5/courriel");

print $mime->get('From')."\n";
print $mime->get('Date')."\n";

my $subject = $mime->get('Subject')."\n";

$subject =~ s/=\?utf-8\?b\?(.*?)\?=/decode_base64($1)/ieg;

print "$subject\n";
```

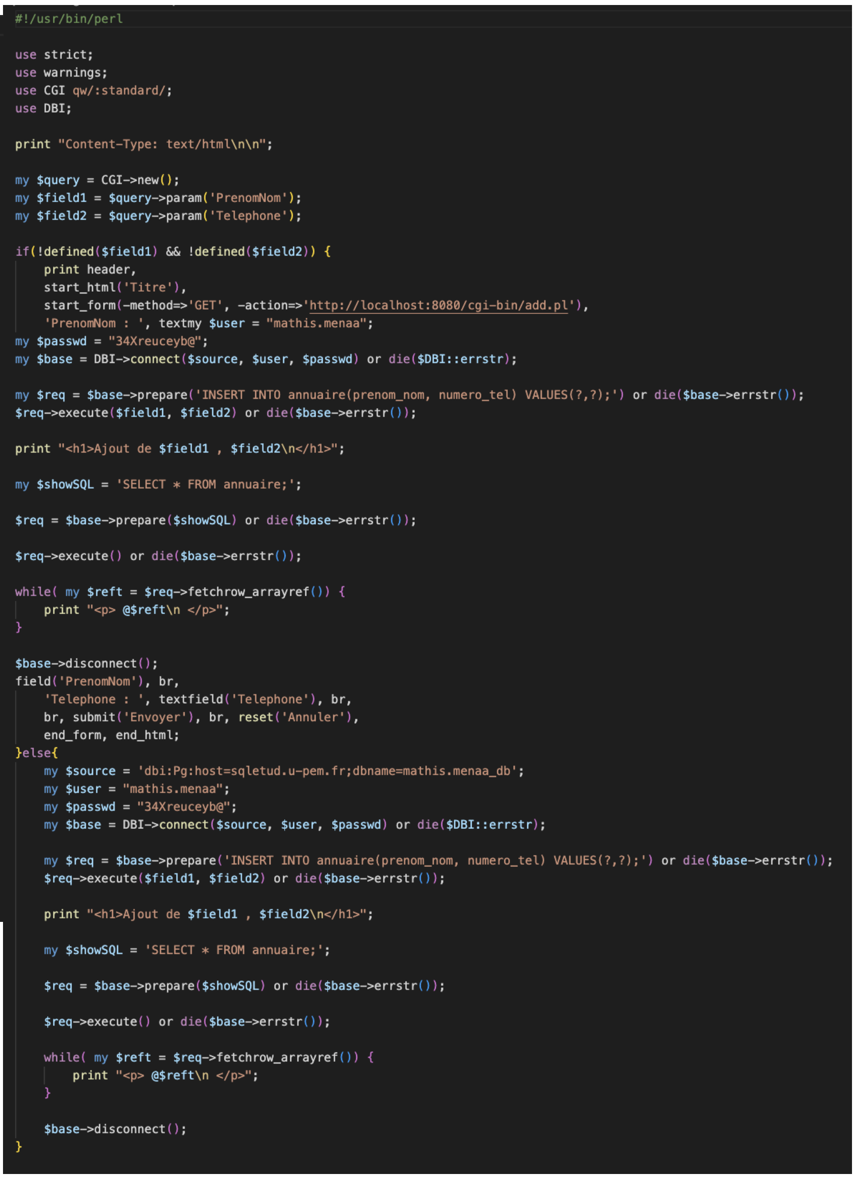


```perl
#!/usr/bin/perl

use strict;
use warnings;
use IO::Socket;
use threads;
use threads::shared;

my $cpt : shared = 1;

my $listen_socket = IO::Socket::INET->new(
    Proto=>'tcp', LocalPort=>2000, Listen=>5, Reuse=>1
    ) or die("$@");

sub sendingFun {
    my ($socketAdress) = @_;
    $socketAdress->send($cpt++."\n");
    sleep(5);
    $socketAdress->send($cpt++."\n");
    close ($socketAdress);
}

while(my $accept_socket = $ listen_socket->accept()) {
    print "New client\n";
    threads->new(\&sendingFun, $accept_socket)->detach();
}
```

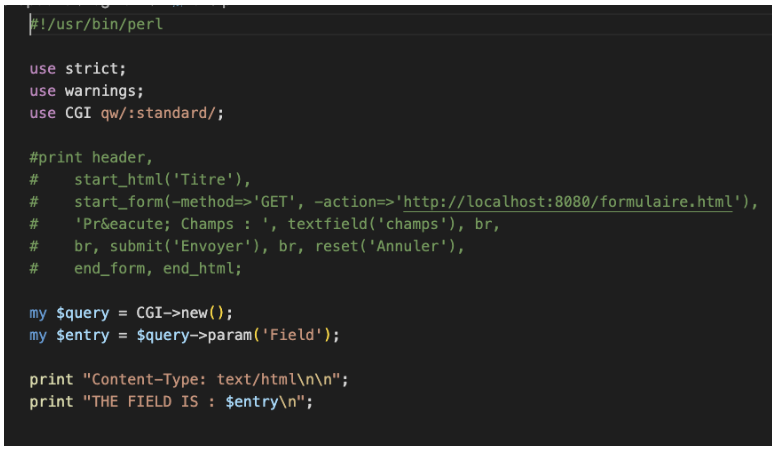
TP6



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

use strict;
use warnings;
use DBI;

my $source = 'dbi:Pg:host=sqletud.u-pem.fr;dbname=mathis.menaa_db';
my $user = "mathis.menaa";
my $passwd = "34Xreuceyb@";
my $base = DBI->connect($source, $user, $passwd) or die($DBI::errstr);

my $createTableSQL =
    'CREATE TABLE annuaire (
        prenom_nom VARCHAR(40),
        numero_tel VARCHAR(20)
    );';

my $req = $base->prepare($createTableSQL) or die($base->errstr());

$req->execute() or die($base->errstr());

my $insertSQL = 'INSERT INTO annuaire(prenom_nom, numero_tel) VALUES(?,?);';

$req = $base->prepare($insertSQL) or die($base->errstr());

$req->execute("Mathis MENAA", "065145574768") or die($base->errstr());
$req->execute("Johan RAMAROSON", "0669454368") or die($base->errstr());
$req->execute("Yohann REGUEME", "065142564") or die($base->errstr());
$req->execute("REMI JR", "07454553468") or die($base->errstr());

my $showSQL = 'SELECT * FROM annuaire;';

$req = $base->prepare($showSQL) or die($base->errstr());

$req->execute() or die($base->errstr());

while( my $reft = $req->fetchrow_arrayref()) {
    print "@$reft\n";
}

$base->disconnect();
```



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

use strict;
use warnings;
use CGI qw/:standard/;
use DBI;

print "Content-Type: text/html\n\n";

my $query = CGI->new();
my $field1 = $query->param('PrenomNom');
my $field2 = $query->param('Telephone');

if(!defined($field1) && !defined($field2)) {
    print header,
    start_html('Titre'),
    start_form(-method=>'GET', -action=>'http://localhost:8080/cgi-bin/add.pl'),
    'PrenomNom : ', textmy $user = "mathis.menaa";
my $passwd = "34Xreuceyb@";
my $base = DBI->connect($source, $user, $passwd) or die($DBI::errstr);

my $req = $base->prepare('INSERT INTO annuaire(prenom_nom, numero_tel) VALUES(?,?);') or die($base->errstr());
$req->execute($field1, $field2) or die($base->errstr());

print "<h1>Ajout de $field1 , $field2\n</h1>";

my $showSQL = 'SELECT * FROM annuaire;';

$req = $base->prepare($showSQL) or die($base->errstr());

$req->execute() or die($base->errstr());

while( my $reft = $req->fetchrow_arrayref()) {
    print "<p> @$reft\n </p>";
}

$base->disconnect();
field('PrenomNom'), br,
    'Telephone : ', textfield('Telephone'), br,
    br, submit('Envoyer'), br, reset('Annuler'),
    end_form, end_html;
}else{
    my $source = 'dbi:Pg:host=sqletud.u-pem.fr;dbname=mathis.menaa_db';
    my $user = "mathis.menaa";
    my $passwd = "34Xreuceyb@";
    my $base = DBI->connect($source, $user, $passwd) or die($DBI::errstr);

    my $req = $base->prepare('INSERT INTO annuaire(prenom_nom, numero_tel) VALUES(?,?);') or die($base->errstr());
    $req->execute($field1, $field2) or die($base->errstr());

    print "<h1>Ajout de $field1 , $field2\n</h1>";

    my $showSQL = 'SELECT * FROM annuaire;';

    $req = $base->prepare($showSQL) or die($base->errstr());

    $req->execute() or die($base->errstr());

    while( my $reft = $req->fetchrow_arrayref()) {
        print "<p> @$reft\n </p>";
    }

    $base->disconnect();
}
```



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

use strict;
use warnings;
use CGI qw/:standard/;

#print header,
#    start_html('Titre'),
#    start_form(-method=>'GET', -action=>'http://localhost:8080/formulaire.html'),
#    'Pr&eacute; Champs : ', textfield('champs'), br,
#    br, submit('Envoyer'), br, reset('Annuler'),
#    end_form, end_html;

my $query = CGI->new();
my $entry = $query->param('Field');

print "Content-Type: text/html\n\n";
print "THE FIELD IS : $entry\n";
```