

```
use strict;
use warnings;

my %cities;

foreach ("Oslo","Trondheim", "Bergen") {
    $cities{$_} = "Norway";
}
foreach ("Stockholm","Malmö", "Kiruna") {
    $cities{$_} = "Sweden";
}

print "Oslo is in $cities{Oslo} and Stockholm in
$cities{Stockholm} you ignorant ----!\n";

my @cities = keys %cities;
print "@cities\n";

my @check = ("Umeå","Malmö");

foreach (@check) {
    if (exists $cities{$_}) {
        print "$_ is stored\n";
    } else {
        print "$_ is not stored\n";
    }
}
```

```
use strict;
use warnings;

my %cities;
foreach ("Oslo","Trondheim","Bergen") {
    $cities{Norway}{$_} = 1;
}

foreach ("Stockholm","Malmö","Kiruna") {
    $cities{Sweden}{$_} = 1;
}

foreach my $country (keys %cities) {
    print "$country:\n";
    foreach my $city (keys %{$cities{$country}}) {
        print "  $city\n";
    }
}

print "Is Kiruna in Norway:";
if (exists $cities{Norway}{Kiruna}) {
    print " YES!\n";
} else {
    print " NO!\n";
}

print "Is Oslo in Norway:";
if (exists $cities{Norway}{Oslo}) {
    print " YES!\n";
} else {
    print " NO!\n";
}

print "Is Stockholm in Finland:";
if (exists $cities{Finland}{Stockholm}) { # Add: exists
    $cities{Finland} &&
    print " YES!\n";
} else {
    print " NO!\n";
}

print "Is Finland stored:";
if (exists $cities{Finland}) {
    print " YES!\n";
} else {
    print " NO!\n";
}
```

```
use strict;
use warnings;

my %cities;
foreach ("Oslo", "Trondheim", "Bergen") {
    $cities{"Norway"}{$_} = 1;
}
foreach ("Stockholm", "Malmö", "Kiruna") {
    $cities{"Sweden"}{$_} = 1;
}

print "Is Kiruna in Norway:";
isin(\%cities, "Kiruna", "Norway");

print "Is Oslo in Norway:";
isin(\%cities, "Oslo", "Norway");

sub isin {

    my $hash = $_[0];
    my $city = $_[1];
    my $country = $_[2];

    if (exists $hash->{$country}{$city}) {
        print " YES\n";
    } else {
        print " NO\n";
    }
}
```

```
use strict;
use warnings;

my @list = @ARGV;

my %hash;
foreach (@list) {
    $hash{$_} ++;
}

#foreach (sort keys %hash) {
#    print "$_: $hash{$_}\n";
#}

my @b = sort keys %hash;

print "@b\n";
```

```
use strict;
use warnings;

my @list1 = (1,2,4,5,6,7);
my @list2 = (1,2,3,8,99);

my %list2;
foreach (@list2) {
    $list2{$_} = 1;
}

my @intersection;
foreach (@list1) {
    push @intersection, $_ if exists $list2{$_};
}

print "@intersection\n";
```

```
use strict;
use warnings;

my @table;

open (E, "microarray.txt");
readline *E; # header
while (<E>) {
    chomp;
    my @row = split /\t/;
    shift @row; # Remove gene name

    push @table, \@row;
}
close (E);

if (@ARGV == 1) {
    if ($ARGV[0] < @table && $ARGV[0] >= 0) {
        print "@{$table[$ARGV[0]]}\n";
    } else {
        print "Invalid index\n";
    }
} elsif (@ARGV == 2) {
    if ($ARGV[0] < @table && $ARGV[0] >= 0 && $ARGV[1] <
@{$table[$ARGV[0]]} && $ARGV[1] >= 0) {
        print "{$table[$ARGV[0]][$ARGV[1]]}\n";
    } else {
        print "Invalid index\n";
    }
} else {
    print "Wrong number of arguments\n";
}
```

```
use strict;
use warnings;

my %tf_module;
my %module_tf;

open (M, "network.txt");
while (<M>) {
    chomp;
    my ($tf,$type,$module) = split /\s/;

    $tf_module{$tf}{$module} = $type;
    $module_tf{$module}{$tf} = $type;
}
close (M);

if (exists $tf_module{$ARGV[0]}) {
    my @m = sort keys %{$tf_module{$ARGV[0]}};
    print "$ARGV[0] regulates: @m\n";
} elsif (exists $module_tf{$ARGV[0]}) {
    my @t = sort keys %{$module_tf{$ARGV[0]}};
    print "$ARGV[0] is regulated by @t\n";
} else {
    print "$ARGV[0] is an unknown tf/module\n";
}
```

```
use strict;
use warnings;

my @table;

open (E, "microarray.txt");
readline *E; # header
while (<E>) {
    chomp;
    my @row = split /\s/;
    shift @row; # Remove gene name

    push @table, \@row;
}
close (E);

print "3 3: $table[3][3]\n";

@table = change_index_by_value(\@table,3,3,500);

print "3 3: $table[3][3]\n";

change_index_by_ref(\@table,3,3,600);

print "3 3: $table[3][3]\n";

sub change_index_by_value {
    my @table = @{$_[0]}; # dereferencing
    my $i      = $_[1];
    my $j      = $_[2];
    my $value  = $_[3];

    $table[$i][$j] = $value;

    return @table;
}

sub change_index_by_ref {
    my $table = $_[0];
    my $i     = $_[1];
    my $j     = $_[2];
    my $value = $_[3];

    $table->[$i][$j] = $value;
}
```

```
use strict;
use warnings;

use Valueref;

my @table;

open (E, "microarray.txt");
readline *E; # header
while (<E>) {
    chomp;
    my @row = split /\s/;
    shift @row; # Remove gene name

    push @table, \@row;
}
close (E);

print "3 3: $table[3][3]\n";

@table = Valueref::change_index_by_value(\@table,3,3,100);

print "3 3: $table[3][3]\n";

Valueref::change_index_by_ref(\@table,3,3,200);

print "3 3: $table[3][3]\n";
```

```
package Valueref;  
  
sub change_index_by_value {  
  my @table = @{$_[0]};  
  my $i     = $_[1];  
  my $j     = $_[2];  
  my $value = $_[3];  
  
  $table[$i][$j] = $value;  
  
  return @table;  
}  
  
sub change_index_by_ref {  
  my $table = $_[0];  
  my $i     = $_[1];  
  my $j     = $_[2];  
  my $value = $_[3];  
  
  $table->[$i][$j] = $value;  
}  
  
1;
```