```
#!/usr/bin/perl
# $Id: haversine.perl,v 1.3 2016-11-08 15:58:49-08 - - $

# Find distance between two airports using the haversine formula.
# http://andrew.hedges.name/experiments/haversine/
# Airport database is in prolog syntax.

use strict;
use warnings;
$0 =~ s|.*/||;

my $PI = 3.14159265358979323846264338327950288419;
my $EARTH_RADIUS_MILES = 3961;

my $database_name = ".score/database.pl";

my %database;
open DATABASE, "<$database_name" or die "$0: $database_name: $!";
while (<DATABASE>) {
   next unless m/airport\(\s*(.*?),\s*'(.*?)',\s*
                  degmin\(\s*(\d+),\s*(\d+)\s*\),\s*
                  degmin\(\s*(\d+),\s*(\d+)\s*\)\s*\)/x;
   my ($airport, $name, $nlatdeg, $nlatmin, $wlondeg, $lonmin)
         = ($1, $2, $3, $4, $5, $6);
   $airport = uc $airport;
   $database{$airport} = [$name, $nlatdeg, $nlatmin, $wlondeg, $lonmin];
}
close DATABASE;

sub radians ($$) {
   # Convert degrees and minutes of arc to radians.
   my ($degrees, $minutes) = @_;
   return ($degrees + $minutes / 60) * $PI / 180;
}

sub print_location(@) {
   my ($deg, $min, $dir) = @_;
   printf " %3d°%2d'%s (%6.2f°, %6.4fr)",
          $deg, $min, $dir, $deg + $min / 60, radians ($deg, $min);
}

sub print_airport($$) {
   my ($airport, $data) = @_;
   printf "%-3s (%-16s)", $airport, $$data[0];
   print_location @$data[1,2], "N";
   print_location @$data[3,4], "W";
   printf "\n";
}

for my $airport (sort keys %database) {
   print_airport $airport, $database{$airport};
}

my $circumference = 2 * $PI * $EARTH_RADIUS_MILES;
printf "Earth radius:        %7.1f miles\n", $EARTH_RADIUS_MILES;
printf "Earth circumference: %7.1f miles\n", $circumference;
printf "Earth 1 degree arc:  %7.1f miles\n", $circumference / 360;
printf "Earth 1 minute arc:  %7.1f miles\n", $circumference / 360 / 60;
```

```
59: printf "Earth 1 radian arc:  %7.1f miles\n", $circumference / $PI / 2;
60:
61: sub haversine_distance ($$$$) {
62:    # Latitude1, longitude1 in radians.
63:    # Latitude2, longitude2 in radians.
64:    my ($lat1, $lon1, $lat2, $lon2) = @_;
65:    my $dlon = $lon2 - $lon1;
66:    my $dlat = $lat2 - $lat1;
67:    my $tmpa = (sin ($dlat / 2)) ** 2
68:             + cos ($lat1) * cos ($lat2) * (sin ($dlon / 2)) ** 2;
69:    my $unit_distance = 2 * atan2 (sqrt ($tmpa), sqrt (1 - $tmpa));
70:    my $distance_miles = $EARTH_RADIUS_MILES * $unit_distance;
71:    return $distance_miles;
72: }
73:
74: while (@ARGV >= 2) {
75:    my $airport1 = shift; $airport1 = uc $airport1;
76:    my $airport2 = shift; $airport2 = uc $airport2;
77:    my $data1 = $database{$airport1};
78:    my $data2 = $database{$airport2};
79:    warn "$0: $airport1, $airport2: invalid airport\n" and next
80:          unless $data1 && $data2;
81:    my $lat1 = radians ($data1->[1], $data1->[2]);
82:    my $lon1 = radians ($data1->[3], $data1->[4]);
83:    my $lat2 = radians ($data2->[1], $data2->[2]);
84:    my $lon2 = radians ($data2->[3], $data2->[4]);
85:    my $distance = haversine_distance ($lat1, $lon1, $lat2, $lon2);
86:    print "\nDistance:\n";
87:    print_airport $airport1, $data1;
88:    print_airport $airport2, $data2;
89:    printf "%.0f miles\n", $distance;
90: }
```