```
#!c:\users\...[your path]...\perl.exe
# this is the path to your installation of PERL on your computer

# Semi-Exhaustive List of Sacred Cuts of the Quadrature of Equilateral Polygons, by Vinyasi.
# This file was originally formed on: 13 Dec. 2003, but was originally discovered using a 1k
RAM PC with monochrome display between 1994 and 1997.
# It was modified on: 7 Dec. 2023.
# No rights reserved. Copy left.
# Retrieved from its previous location of:
http://vinyasi.mayashastra.org/book/zip/sacred_square.zip
# Located at Archive.org:
https://web.archive.org/web/20051029163755/http://vinyasi.mayashastra.org/book/zip/sacred_squa
re.zip
# Uploaded to: http://vinyasi.info/Infinite%20Range%20of%20Golden%20Ratios/original%20research
%20during%201994%20to%201997/sacred_cuts_v2c_save.txt

# HINT ...
# COMMENT OUT THE DEBUG LINES IF YOU DON'T WANT EXPLANATORY TEXT WRITTEN TO A FILE

$iwannasearch = 3; # 13 is the maximum length of digits among primes that I may search for
# Hint: 3 digits means that primes less than 1000 will be my upper limit
# Hint: I have never been able to find any prime greater than the integer of 5 to satisfy
these search criteria.
# This is not five digits, but the number '5', as in how many fingers do I have on one hand?

#$kill_debug = 7; # at what prime is it appropriate to kill a runaway test?
# Hint: 7 is good.

################################################################################
## DO NOT CHANGE ANYTHING BENEATH THIS LINE UNLESS YOU KNOW WHAT YOU ARE DOING ##
################################################################################

$truelimit = 15; # this is how many digits of accuracy which are available to my computer.
this is the total which includes both sides of the decimal point.
$marginForError = 0; # how many extra digits to allow for accuracy

# bailout if ...
if($marginForError < 0) {
      die "\nERROR!\n" .
      "$marginForError MUST BE GREATER THAN OR EQUAL TO ZERO\n" .
      "TRY SOMETHING LARGER.\n$!"; }

$mylimit = 1; # 1; # this limit is less than one-half of how many digits I must have access to
on my computer for accuracy to succeed
$bailout = int(($truelimit - $marginForError - $iwannasearch) / 2);
$oops = $truelimit - ($mylimit * 2) - $marginForError;

# bailout if ...
if($bailout < $mylimit or $iwannasearch < 1) {
      die "\nERROR!\n" .
      "TOO MANY DIGITS.\n" .
      "THE LIMIT IS $oops DIGIT(S).\n" .
      "BUT YOU WANT TO SEARCH FOR $iwannasearch DIGITS.\n" .
      "THAT WON'T WORK!\nTRY SOMETHING SMALLER.\n$!"; }

$safelimit = $truelimit - $marginForError - $iwannasearch; # accuracy of search to this many
digits

# bailout if ...
if($safelimit < 0) {
      die "\nERROR!\n" .
      "\$safelimit = $safelimit MUST BE GREATER THAN OR EQUAL TO ZERO\n" .
      "TRY SOMETHING LARGER.\n$!"; }

$shift = int(10 ** ($safelimit / 2)); # round-down to the nearest integer
$limit = 10 ** $iwannasearch; # maximum numeric value to search for primes which meet the
```

```
criteria for sacred cuts

$correct_data = "correct_data.txt"; # NAME OF TEXT FILE TO WRITE SUCCESSFUL OUTCOMES
$raw_data = "raw_data.txt"; # NAME OF TEXT FILE TO WRITE EXPLANATORY TEXT # UNCOMMENT TO DEBUG

open(OUTPUTDATA, ">", $correct_data) or die "open $correct-data failed: $!"; # use ">" for
erasing the file and rewriting to it
open(OUTPUTCOMMENTS, ">", $raw_data) or die "open $raw-data failed: $!"; # use ">>" for
appending to the file or use ">" for erasing the file and rewriting to it # UNCOMMENT TO DEBUG

print OUTPUTDATA "Sacred Square Cuts Among Even-Sided Polygons", "\n\n";

use constant PI => 4 * atan2 1, 1;

$printMsgOfProgress = 1; # UNCOMMENT TO DEBUG
$prynt = 1;
$start = 2; # minimum search value for primes

print OUTPUTDATA ">> Search range is from ", $start, " to ", $limit, " <<\n\n";
print OUTPUTCOMMENTS "BEGIN DEBUGGING OUTPUT\n"; # UNCOMMENT TO DEBUG

for($x1 = $start; $x1 <= $limit; $x1++) { # even sides
print OUTPUTCOMMENTS "\n\n\$x1 = $x1 >> FOR LOOP >> THIS IS SUPPOSED TO BE A PRIME NUMBER\n";
# UNCOMMENT TO DEBUG
  $printMsgOfProgress = 1; # UNCOMMENT TO DEBUG
  $prynt = 1;
  $mod = "false";
  for($x2 = $start; $x2 <= sqrt($x1); $x2++) {
print OUTPUTCOMMENTS "  \$x2 = $x2 >> FOR LOOP >> $x2 IS LESS THAN OR EQUAL TO THE SQUARE ROOT
OF $x1\n"; # UNCOMMENT TO DEBUG
    if($x1 % $x2 == 0) {
print OUTPUTCOMMENTS "    ", $x1 % $x2, " = $x1 MOD $x2 >> THE TEST FOR PRIMALITY HAS FAILED
FOR THE COMPOSITE NUMBER OF: $x1.\n    I AM SKIPPING TO THE NEXT INTEGER ...\n"; # UNCOMMENT
TO DEBUG
      $mod = "true";
      last; }}

  if($mod eq "false") {
    $read = "";
    $tot_sides = $x1 * 4; # even sides
print OUTPUTCOMMENTS "      $tot_sides = $x1 TIMES 4 >> QUADRATURE OF SIDES\n"; # UNCOMMENT TO
DEBUG
    $readout = "\n". $x1. "p ". $tot_sides. "-Gon\n";
    $num_ang = $tot_sides / 2; # even sides
print OUTPUTCOMMENTS "        $num_ang = $tot_sides DIVIDED BY 2 >> NUMBER OF ANGLES TO
COMPUTE\n"; # UNCOMMENT TO DEBUG
    $ang = 360 / $tot_sides; # even sides
print OUTPUTCOMMENTS "          $ang DEGREES >> SMALLEST ANGLE TO COMPUTE\n"; # UNCOMMENT TO
DEBUG
    for($x3 = 1; $x3 <= $num_ang; $x3++) { # even sides
      $angle = $ang * $x3;
#print OUTPUTCOMMENTS "            $angle = $ang TIMES $x3 >> \$x3 FOR LOOP >> USE THIS ANGLE
OF $angle DEGREES\n"; # UNCOMMENT TO DEBUG
      $sin[$x3] = 2 * sin(PI * $angle / 360); # even sides
#print OUTPUTCOMMENTS "              $sin[$x3] = 2 TIMES sin(PI DIVIDED BY (", $tot_sides /
$x3, " = $tot_sides DIVIDED BY $x3)) >> CALCULATE THE UNIT LENGTH OF CHORD #$x3 IN A CIRCLE OF
ONE UNIT RADIUS\n"; # UNCOMMENT TO DEBUG
print OUTPUTCOMMENTS "              CHORD #$x3\n"; # UNCOMMENT TO DEBUG
      $readout .= "Angle No.". $x3. ", {Sin(". (int(int($angle * $shift + 1) / $shift));
      $readout .= "Â° Ã· 2)} x 2 = ". $sin[$x3]. "\n"; }
#print OUTPUTCOMMENTS "\n                THE FOLLOWING COLUMNS OF NUMBERS MUST BE NON-ZERO
RATIONAL INTEGERS OR FRACTIONS TO SATISFY THE COEFFICIENTS OF A QUADRATIC POLYNOMIAL.\n"; #
UNCOMMENT TO DEBUG
      for($x4 = 1; $x4 <= $num_ang; $x4++) { # even sides
        for($x5 = ($x4 + 1); $x5 <= $num_ang; $x5++) { # even sides
```

```
          for($x7 = 1; $x7 <= $num_ang; $x7++) { # even sides
          $prynt = 1;
            for($x8 = ($x7 + 1); $x8 <= $num_ang; $x8++) { # even sides
              for($alt_sign = -1; $alt_sign <= 1; $alt_sign += 2) {
                $bsign = "- ";
                $csign = "+ ";
                $sin2sign = "+ ";
            for($alt1 = -1; $alt1 <= 1; $alt1 += 2) {
              for($alt2 = -1; $alt2 <= 1; $alt2 += 2) {
                $alt3 = $alt1 * (-1);
                $alt4 = $alt2 * (-1);
                $al1 = '';
                $al2 = '';
                $al3 = '';
                $al4 = '';
                $al1 = "<sup>-</sup>" if($alt1 < 0);
                $al2 = "<sup>-</sup>" if($alt2 < 0);
                $al3 = "<sup>-</sup>" if($alt3 < 0);
                $al4 = "<sup>-</sup>" if($alt4 < 0);
                $sin1 = $sin[$x4] ** ($alt1) * $sin[$x5] ** ($alt3);
                $sin2 = $sin[$x7] ** ($alt2) * $sin[$x8] ** ($alt4);
                $be = $sin1 + ($alt_sign * $sin2);
                $ce = $alt_sign * $sin1 * $sin2;
                if($alt_sign < 0) {
                      $sin2 = abs($sin2);
                      $sin2sign = "- "; }
                    if($be < 0) {
                      $be = abs($be);
                      $bsign = "+ "; }
                    if($ce < 0) {
                      $ce = abs($ce);
                      $csign = "- "; }
#print OUTPUTCOMMENTS "\n                  +$ae $bsign$be $csign$ce"; # UNCOMMENT TO DEBUG
#if($x1 > $kill_debug) { die } # UNCOMMENT TO DEBUG
                if($be != 0 and length $be < $safelimit and length $ce < $safelimit and
(length $sin1 >= $safelimit or length $sin2 >= $safelimit)) {
                  if(!$read) {
print OUTPUTCOMMENTS "\n\n>>> SUCCESS! GATHER UP ALL OF THE INFORMATION AND SEND IT TO A TEXT
FILE WITH THE FILE-NAME OF: $correct_data\n"; # UNCOMMENT TO DEBUG
                    $read = "yes";
                    print OUTPUTDATA $readout, "\n"; }
                    $ae = 1;
                    if($ce =~ /\./) {
                    $ae *= 1 / $ce;
                    $be *= 1 / $ce;
                    $ce *= 1 / $ce; }
                  if($prynt == 1) {
                    $prynt = 0;
                        print OUTPUTDATA "When the reciprocal of Angle No.", $x4, " (", 1 /
$sin[$x4], ") is multiplied by Angle No.", $x5, " (", $sin[$x5], "), then this equals the
length of a diagonal: ";
                        print OUTPUTDATA $sin1, ".";
                        print OUTPUTDATA "\nLikewise, when Angle No.", $x7, " (", $sin[$x7], ")
is multiplied by the reciprocal of Angle No.", $x8, " (", 1 / $sin[$x8], "), then this yields
the length of another diagonal: ";
                        print OUTPUTDATA $sin2sign, $sin2, ".";
                        print OUTPUTDATA "\nAnd when the first diagonal is divided by the second
diagonal, and when the second diagonal is divided by the first diagonal, then this yields the
two roots of a quadratic polynomial:\n";
                        print OUTPUTDATA "{", $sin1, ", ", $sin2sign, $sin2, "}", " = ";
                    print OUTPUTDATA $ae if($ae != 1);
                    print OUTPUTDATA "x^2 ", $bsign;
                    print OUTPUTDATA $be if($be ne '1');
                    print OUTPUTDATA "x ", $csign, $ce, "\n\n"; }
                  } else {
```

```
if($printMsgOfProgress == 1) { # UNCOMMENT TO DEBUG
$printMsgOfProgress = 0; # UNCOMMENT TO DEBUG
print OUTPUTCOMMENTS "\n\n>>> HAVE EXHAUSTED SEARCHING FOR ANY MORE POLYNOMIALS OF INTEGER
COEFFICIENTS FOR THE PRIME NUMBER OF $x1.\n"; # UNCOMMENT TO DEBUG
print "\nI HAVE FINISHED SEARCHING FOR POLYNOMIALS OF INTEGER\nCOEFFICIENTS WITHIN EQUILATERAL
POLYGONS WHOSE NUMBER\nOF SIDES EQUALS FOUR TIMES THE PRIME NUMBER OF: $x1.\nI WILL CONTINUE
TO SEARCH UNTIL I REACH THE LIMIT OF $limit.\n"; } # UNCOMMENT TO DEBUG
            }}}}}}}}
    undef @sin; }}

print OUTPUTCOMMENTS "\n\nEND DEBUGGING OUTPUT\n"; # UNCOMMENT TO DEBUG

close OUTPUTDATA;
close OUTPUTCOMMENTS; # UNCOMMENT TO DEBUG

exit;
```