```
//		8/30/2010 Justin Coslor
//		Prototype Code for Prime Number Spacing Midpoint Divisibility Test
//
//		3 + 4n means spacings of odd odds (aka soo), we will test if they have midpoints divisible by 3
//		5 + 4n means spacings of even odds (aka seo), we will test if they have midpoints divisible by 2
//		11 + 12n means spacing of void odd odds (aka svoid), we will test if they have midpoints that have
//		no common divisors.
//
//		Goal: to test for the next potential prime from a known prime using the patterns above.
//
//		Given:
//		s = soo or seo or svoid = spacing from one known prime to the next potential prime
//		p1 = known prime number
//		p2 = next potentially prime number after p1
//		p2 = (p1 + s)
//		m = the midpoint location inside each spacing between a known prime and a potential prime.
//		divisors = all prime divisors of each midpoint (use you old divisor code)

p1 = 3		// p1 starting value
p2 = 5		// p2 starting value
m = (p1 + p2) / 2	// midpoint equation gets updated and printed by the for loop below
divisors = m / ? % 0	// use your old code to catalog and print midpoint prime divisors

for n = 0 to p1^2		// 0 to p1^exponent is how many examples to test.

		soo = 3 + 4n		// definition of soo
		seo = 5 + 4n		// definition of seo
		svoid = 11 + 12n	// definition of svoid
		m = (p1 + p2) / 2	// definition of midpoint

		if (p1 + (p1 + soo)) / 2 = 3 % 0
				// odd odd spacing test --- (p1 + soo) = p2, so ((p1 + p2) / 2) = m. Is m divisible by 3?
		then p2 = (p1 + soo)		// if m is divisible by 3 then set p2 to the tested spacing ahead
		print p1, p2, m, soo, n, divisors, "soo success, p2 might equal ", p2
		p1 = p2			// set p1 to p2 to move the next prime test ahead for the next n

		else if (p1 + (p1 + seo)) / 2 = 2 % 0
				// even odd spacing test --- (p1 + seo) = p2, so ((p1 + p2) / 2) = m. Is m divisible by 2?
		then p2 = (p1 + seo)			// if m is divisibly by 2 then set p2 to the tested spacing ahead
		print p1, p2, m, seo, n, divisors, "seo success, p2 might equal ", p2
		p1 = p2			// set p1 to p2 to move the next prime test ahead for the next n

		else p2 = (p1 + svoid)		// odd odd svoid spacing test --- (p1 + svoid) = p2. When we print m's
						// divisors it will show that there are no common divisors from one svoid
						// instance of n in the for loop to another.
		print p1, p2, m, svoid, n, divisors, "svoid occurance, p2 unknown"
		p1 = p2			// set p1 to p2 to move the next prime test ahead for the next n

		endif
repeat

//		I was unsure how to get the for loop to display potential prime numbers sequentially,
//		maybe a solution would be to write three programs, one for soos, one for seos,
//		and one for svoids.
```