```
> #begin initialization section
> # it is nice to keep initialization simple
> # find primes that have neighbors that fit a pattern p
>
> m := 30030 :
> o := [1271, 4001, 5891, 8621, 12821, 15131, 15551, 17861, 19751, 22481] :
> p := [0, 2, 6, 8, 12, 18, 20, 26, 30, 32] :
>
> # end initialization section
>
>
>
> # produce a large integer that is the product of some small primes.
> a := 1 :
  for b from 1 to 25 do
  a := a·ithprime(b) :
  end do:
  a;
```

$$230556796394551842475310214733175607 0 \quad (1)$$

```
> # now 'a' is the product of the primes less than 100.
> composite_small :=proc(n :: integer)
  description "determine if n has a prime factor less than 100"
  if igcd(230556796394551842475310214733175607 0, n) = 1 then
  return false
  else return true;
  end if;
  end proc:
> composite_small(12)
```

$$true \quad (2)$$

```
> composite_small(101)
```

$$false \quad (3)$$

```
> #so composite_small tests if there are any factors 2 through 97.
>
> #A prime constellation pattern of length given length
>
> 2·3·5·7·11
```

$$2310 \quad (4)$$

```
> # consider m 2310 is 2·3·5·7·11  so primeNext is now 11, the next prime.
> primeNext := 17 :
>
>
>
> # using isprime(m·n + o + p)
>
> with(ArrayTools) :
> os := Size(o, 2);
```

(5)

```
os := 10                                                    (5)
> ps := Size(p, 2);
ps := 10                                                    (6)
>
>
> # ps stands for pattern size
> # begin refinement section
> # the refinement section is not complete.
> #it would be cool if I could automate the process of lengthening the p vector
> #maybee I can colaborate with someone on this
>
>
> #end refinement section
>
>
>
> loopstart := 0 :
> loopstop := 10^11 :
>
>
>
>
> print(11);
11                                                          (7)
> # my composite-small procedure incorrectly treats 11 as a composite number.
>
>
> for n from loopstart to loopstop do
  for a from 1 to os do
  counter := 0 :
  FirstCounter := 0 :
  SecondCounter := 0;
  while SecondCounter > -10 and SecondCounter < ps do
  SecondCounter := SecondCounter + 1;
  if composite_small(m·n + o[a] + p[SecondCounter]) = false then SecondCounter
       := SecondCounter + 1; else SecondCounter := -10; end if;
  end do;
  wc := 0 :
  if SecondCounter ≥ 9 then
  while counter ≥ 0 and FirstCounter < ps do

  wc := wc + 1;
  if isprime(m·n + o[a] + p[counter + 1]) then counter := counter + 1 else counter := -1;
       end if;
  end do;
    end if;
```

```
    if counter = ps then print(m·n + o[a]) end if;
     end do:
    end do:
Warning,  computation interrupted
> counter
                                  0                                  (8)
> n
                                350300                               (9)
> a
                                  4                                  (10)
> FirstCounter
                                  0                                  (11)
> m·n
                              10519509000                            (12)
>
```