Pseudocode

```
PROMPT for n
GET n

prime <- [TRUE FOR i in range(n+1)]          # A

p <- 2                                       # B
WHILE (p * p <= n)                           # C
     IF (prime[p] IS TRUE)                   # D
          FOR i in range(p * p, n + 1, p)    # E
               prime[i] <- FALSE             # F
     p <- p + 1                              # G

primes <- []
FOR p in range(2, n)
     IF prime[p]
          primes.add(p)


PUT primes
```

Program Trace

	prime	p	n	i
A	[T, T, T, T, T, T, T, T, T, T, T]	/	10	/
B	[T, T, T, T, T, T, T, T, T, T, T]	2	10	/
C	[T, T, T, T, T, T, T, T, T, T, T]	2	10	/
D	[T, T, T, T, T, T, T, T, T, T, T]	2	10	/
E	[T, T, T, T, T, T, T, T, T, T, T]	2	10	2
F	[T, T, T, T, F, T, F, T, F, T, F]	2	10	2
E	[T, T, T, T, F, T, F, T, F, T, F]	2	10	4
F	[T, T, T, T, F, T, F, T, F, T, F]	2	10	4
G	[T, T, T, T, F, T, F, T, F, T, F]	3	10	/
C	[T, T, T, T, F, T, F, T, F, T, F]	3	10	/
D	[T, T, T, T, F, T, F, T, F, T, F]	3	10	/
E	[T, T, T, T, F, T, F, T, F, T, F]	3	10	9
G	[T, T, T, T, F, T, F, T, F, T, F]	4	10	/

Algorithmic Efficiency

This algorithm is O(log n). While there is a loop inside of another loop, neither of the loops run through every iteration. On top of that, the nested loop does not even fire every time. As the input size increases, the performance is a log to the algorithm.