isPrime[n]: returns True if n is prime; False otherwise

Print: displays output on the screen

Return: sends a value back to whatever called the module, useful for future computation

Implementation 1:

for each $k = 2, 3, \ldots, \lfloor \sqrt{n} \rfloor$, check if k divides n

```
isPrime[n_Integer] := Module[{},
  Do[
   If[Divisible[n, k], Return[False, Module]],
   {k, 2, Sqrt[n]}];
  Return[True]
 ]
```

necessary to make Mathematica quit the module, not just the loop

only runs if no divisor is found

Implementation 2:

test divisor

```
isPrime2[n_] := Module[ {foundDivisor = False, d = 2, m = Sqrt[n]},
  While[! foundDivisor && d ≤ m,
   If[Divisible[n, d], foundDivisor = True];
   d++
  ];

  ! foundDivisor (* return value *)
 ]
```

not — and — condition

add 1 to d

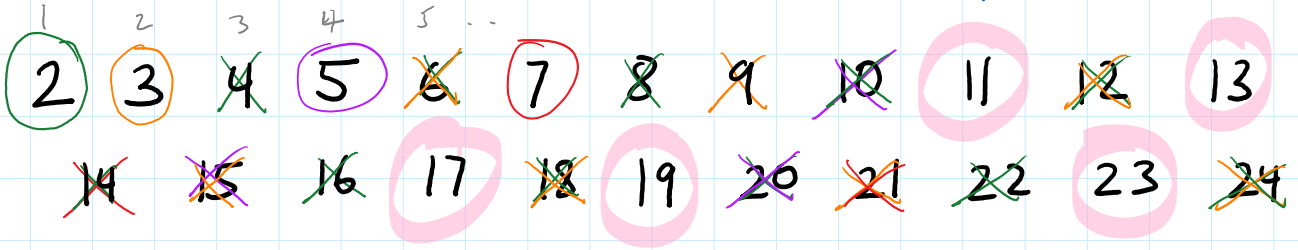
Implementation 3:

```
isPrime3[n_Integer] := Module[{m = Sqrt[n]},
  Catch[
   Do[
    If[Divisible[n, d], Throw[False]],
    {d, 2, m}
   ];
   True
  ] (* the Module returns the "caught" value *)
 ]
```

return value if no divisor is found

transfers control of the program to the enclosing Catch statement, which evaluates to False

SIEVE OF ERATOSTHENES: How to implement?



option 1: start with list 2, 3, ..., n

delete multiples of 2,
then multiples of 3, etc.

Delete

option 2: when we find a non-prime, don't delete, but set that list entry to False or 0 or something