Part I – A Brief Introduction to Mathematica Modules

Syntax: `Module[{local variables}, body]` where `body` is a single Mathematica statement or a sequence of Mathematica statements separate by semi-colons

The value of the last statement is the value of `Module`

Syntax: `If[condition, result_1, result_2]` where `result_1` and `result_2` are single Mathematica statements or sequences of Mathematica statements separated by semi-colons

Syntax: `While[condition, body]` where `body` is a single Mathematica statement or a sequence of Mathematica statements separate by semi-colons

Syntax: `Print[ expression_1, expression_2, ...]`

Syntax: `For[init-stmts, conditions, changes, body]` where `body` is a single Mathematica statement or a sequence of Mathematica statements separate by semi-colons

For executes the initialization statements, evaluates the conditions and as long as they are true executes the `body` and the change statements.

Implement and test these examples

Example 1: A function to sum the integers from 1 to n

```mathematica
f[n_] := Module[{k, sum},
  sum = 0;
  For[k = 1, k <= n, k = k + 1,
  sum = sum + k];
  sum]
```
Example 2: a function to detect primes

```mathematica
isPrime[n_] := Module[{k},
  k = 2;
  While[Mod[n, k] != 0,
    k = k + 1;
  ]
  If[k == n,
    Print[n, " is Prime"],
    Print[k, " divides ", n, " - not Prime"]
  ];
]
```

Example 3: another function to detect primes

```mathematica
isPrime2[n_] := Module[{k},
  For[k = 2,
    k <= n && Mod[n, k] != 0,
    k++;
  ]
  If[k == n,
    Print[n, " is Prime"],
    Print[k, " divides ", n, " - not Prime"]
  ];
]
```

**Part II** – Write a Mathematica module that finds a zero using the bisection method; that is complete the following code

```mathematica
Bisection[f_, a0_, b0_, t0_] := Module[{ },
  while ( b - a > tolerance)
    midPt = (a + b) /2
    if f(a) * f(midPt) < 0
      b = midPt
    else
      a = midPt
    return (a + b)/2, f((a + b)/2)
]
```

**Part III.** – Create your own version of the Page293Example3.nb notebook