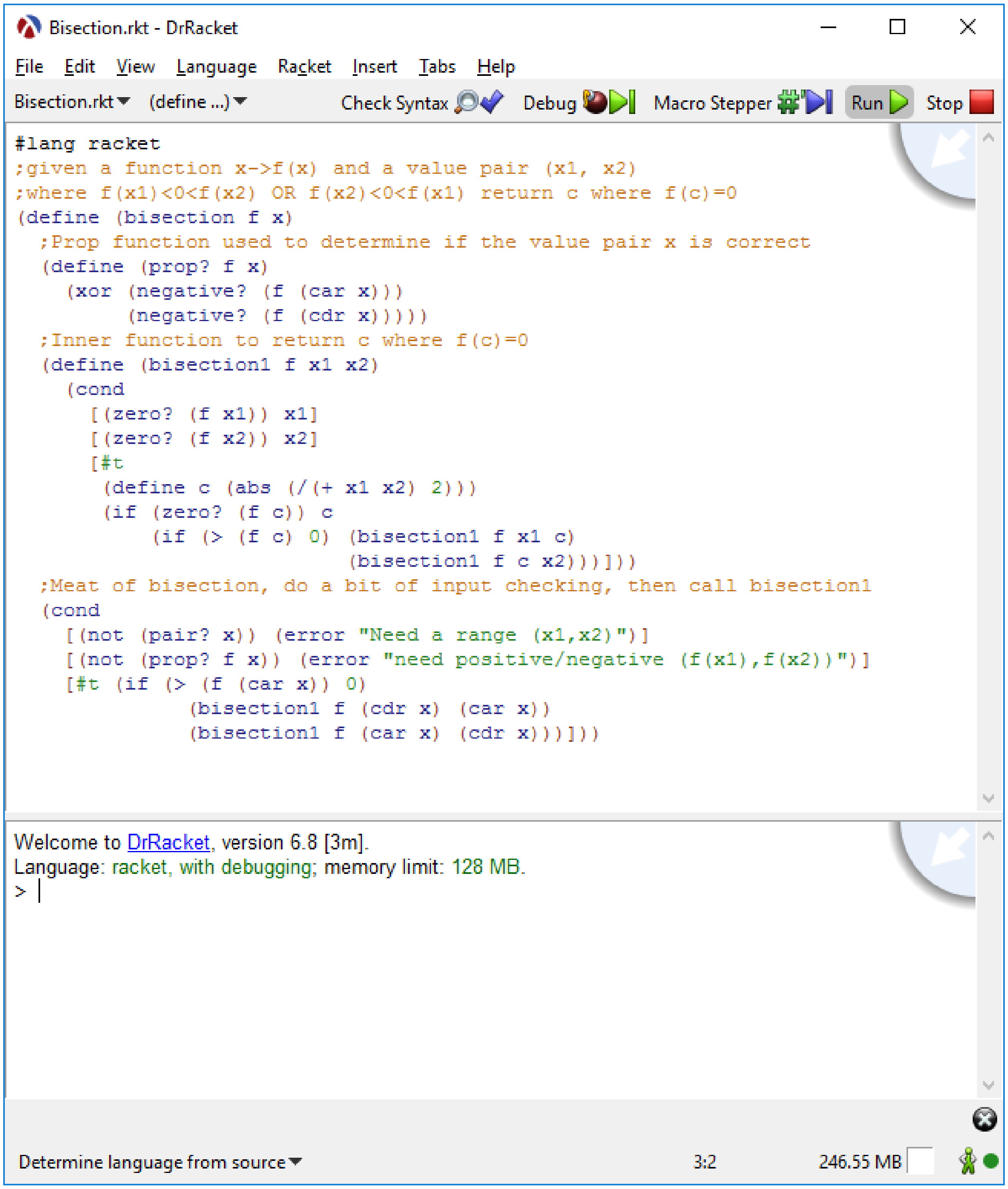
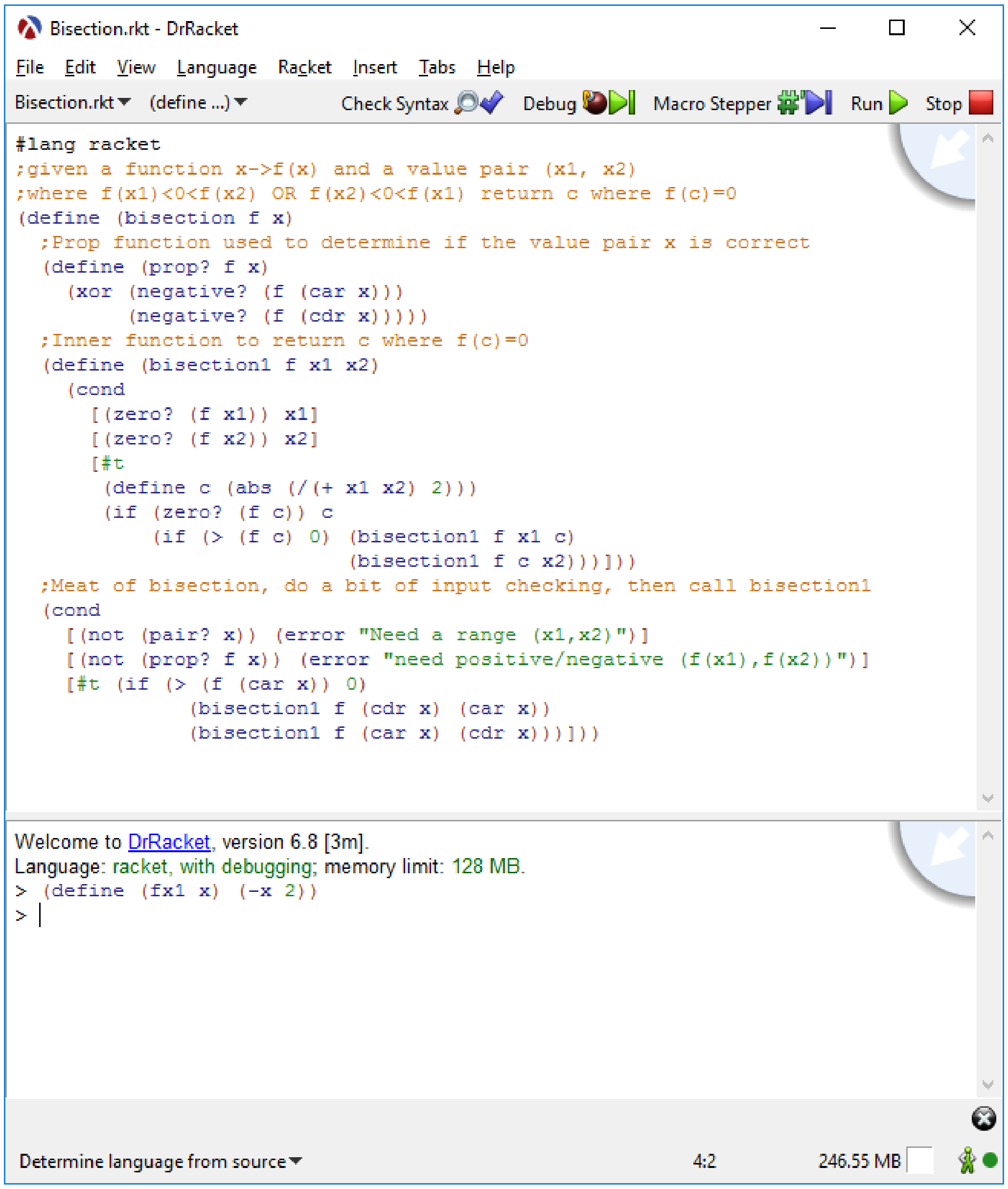
1. Download the project
2. Browse to the download
3. Install racket https://download.racket-lang.org/
4. Double click Bisection.rkt
5. Click Run


Bisection.rkt - DrRacket
File Edit View Language Racket Insert Tabs Help
Bisection.rkt (define ...) Check Syntax Debug Macro Stepper Run Stop

```
#lang racket
;given a function x->f(x) and a value pair (x1, x2)
;where f(x1)<0<f(x2) OR f(x2)<0<f(x1) return c where f(c)=0
(define (bisection f x)
  ;Prop function used to determine if the value pair x is correct
  (define (prop? f x)
    (xor (negative? (f (car x)))
         (negative? (f (cdr x)))))
  ;Inner function to return c where f(c)=0
  (define (bisection1 f x1 x2)
    (cond
      [(zero? (f x1)) x1]
      [(zero? (f x2)) x2]
      [#t
       (define c (abs (/(+ x1 x2) 2)))
       (if (zero? (f c)) c
           (if (> (f c) 0) (bisection1 f x1 c)
                           (bisection1 f c x2)))]))
  ;Meat of bisection, do a bit of input checking, then call bisection1
  (cond
    [(not (pair? x)) (error "Need a range (x1,x2)")]
    [(not (prop? f x)) (error "need positive/negative (f(x1),f(x2))")]
    [#t (if (> (f (car x)) 0)
              (bisection1 f (cdr x) (car x))
              (bisection1 f (car x) (cdr x)))]))
```

```
Welcome to DrRacket, version 6.8 [3m].
Language: racket, with debugging; memory limit: 128 MB.
> 
```

6. Now, go into the bottom half of the program, this section is called the REPL, it allows you to run code without compiling.
   a. Go for it, try typing (+ 4 5) and hit enter, or (/ 3 4)
7. Now you can sample code, this file provies a “bisection” function, which requires a function that takes x and returns y (x->y). First thing you will have to do is come up with or create one, here are some examples
   a. (define (fx1 x) (- x 2))
   b. (define (fx2 x) (/ x 5))
   c. (sin)
   d. Define some of them in the REPL

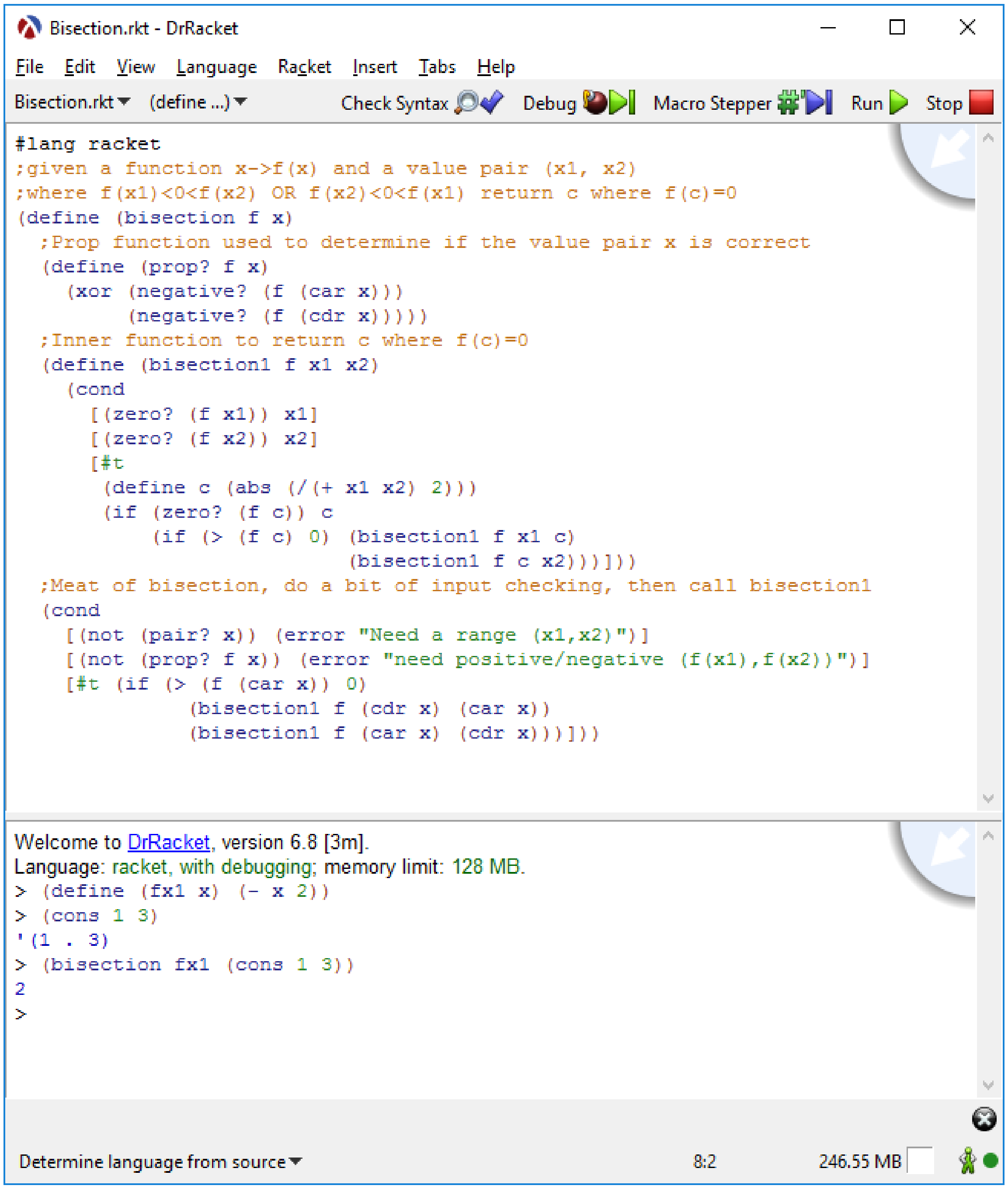

Bisection.rkt - DrRacket
File Edit View Language Racket Insert Tabs Help
Bisection.rkt (define ...)
Check Syntax
Debug
Macro Stepper
Run
Stop

```
#lang racket
;given a function x->f(x) and a value pair (x1, x2)
;where f(x1)<0<f(x2) OR f(x2)<0<f(x1) return c where f(c)=0
(define (bisection f x)
  ;Prop function used to determine if the value pair x is correct
  (define (prop? f x)
    (xor (negative? (f (car x)))
         (negative? (f (cdr x)))))
  ;Inner function to return c where f(c)=0
  (define (bisection1 f x1 x2)
    (cond
      [(zero? (f x1)) x1]
      [(zero? (f x2)) x2]
      [#t
       (define c (abs (/(+ x1 x2) 2)))
       (if (zero? (f c)) c
           (if (> (f c) 0) (bisection1 f x1 c)
                           (bisection1 f c x2)))]))
  ;Meat of bisection, do a bit of input checking, then call bisection1
  (cond
    [(not (pair? x)) (error "Need a range (x1,x2)")]
    [(not (prop? f x)) (error "need positive/negative (f(x1),f(x2))")]
    [#t (if (> (f (car x)) 0)
            (bisection1 f (cdr x) (car x))
            (bisection1 f (car x) (cdr x)))]))
```

```
Welcome to DrRacket, version 6.8 [3m].
Language: racket, with debugging; memory limit: 128 MB.
> (define (fx1 x) (-x 2))
> |
```

8. Now you can run “bisection”
   a. (bisection fx1 (cons 1 3))

b. The “bisection” function requires a value pair, represented as ‘(1 . 3) this can be made with (cons 1 3)… try it



```
#lang racket
;given a function x->f(x) and a value pair (x1, x2)
;where f(x1)<0<f(x2) OR f(x2)<0<f(x1) return c where f(c)=0
(define (bisection f x)
  ;Prop function used to determine if the value pair x is correct
  (define (prop? f x)
    (xor (negative? (f (car x)))
         (negative? (f (cdr x)))))
  ;Inner function to return c where f(c)=0
  (define (bisection1 f x1 x2)
    (cond
      [(zero? (f x1)) x1]
      [(zero? (f x2)) x2]
      [#t
       (define c (abs (/(+ x1 x2) 2)))
       (if (zero? (f c)) c
           (if (> (f c) 0) (bisection1 f x1 c)
                           (bisection1 f c x2)))]))
  ;Meat of bisection, do a bit of input checking, then call bisection1
  (cond
    [(not (pair? x)) (error "Need a range (x1,x2)")]
    [(not (prop? f x)) (error "need positive/negative (f(x1),f(x2))")]
    [#t (if (> (f (car x)) 0)
              (bisection1 f (cdr x) (car x))
              (bisection1 f (car x) (cdr x)))]))
```

```
Welcome to DrRacket, version 6.8 [3m].
Language: racket, with debugging; memory limit: 128 MB.
> (define (fx1 x) (- x 2))
> (cons 1 3)
'(1 . 3)
> (bisection fx1 (cons 1 3))
2
>
```