CSE320 Lab Report .Scheme

Result for quicksort

```
[006704029@csusb.edu@csevnc lan3Scheme]$ time guile -s qsscheme.sch
;;; note: source file /home/csusb.edu/006704029/CSE320/lan3Scheme/qsscheme.sch
;;;       newer than compiled /home/csusb.edu/006704029/.cache/guile/ccache/2.0-LE-8-2.0/home/csusb
edu/006704029/CSE320/lan3Scheme/qsscheme.sch.go
;;; note: auto-compilation is enabled, set GUILE_AUTO_COMPILE=0
;;;       or pass the --no-auto-compile argument to disable.
;;; compiling /home/csusb.edu/006704029/CSE320/lan3Scheme/qsscheme.sch
;;; compiled /home/csusb.edu/006704029/.cache/guile/ccache/2.0-LE-8-2.0/home/csusb.edu/006704029/CS
320/lan3Scheme/qsscheme.sch.go
unsorted:
#(11 22 44 33 77 66 55 99)
sorted:
#(11 22 33 44 55 66 77 99)

unsorted:
#(112 28 81 198 92 466 58 597 46 989)
sorted:
#(28 46 58 81 92 112 198 466 597 989)

unsorted:
#(222 444 587 456 455 678 716 729 782 239 195 495 794 309 988)
sorted:
error one of these: low or high < 0
#(195 222 239 309 444 455 456 495 587 678 716 729 782 794 988)

unsorted:

real    0m0.138s
user    0m0.133s
sys     0m0.016s
```

Source code

```
(define ar (vector 11 22 44 33 77 66 55 99))
(define size (vector-length ar))

(define ar2 (vector 112 28 81 198 92 466 58 597 46 989))
(define size2 (vector-length ar2))

(define ar3 (vector 222 444 587 456 455 678 716 729 782 239 195 495 794 309 988))
(define size3 (vector-length ar3))

(define partition
  (lambda (low high)
    ;(display "partition\n")
    (let ((pivot (vector-ref ar high)) (i (- low 1)) (j low))
      ;(display "start of loop: ")
      ;(display i)
      ;(display " ")
      ;(display j)
      ;(display "\n")
      (while (< j high)
        ;(display "loop\n")
        (if (< (vector-ref ar j) pivot)
          (begin
            ;(display "element is smaller\n")
            (set! i (+ i 1))
            ;(display "i is incremented: ")
            ;(display i)
            ;(display "\n")
            (let ((temp 0))   ;swap ar[i+1] with ar[high]
                ;(display "temp is ")
                ;(display temp)
                ;(display "\n")
              (set! temp (vector-ref ar i))
                ;(display "temp is now ")
                ;(display temp)
                ;(display "\n")
              (vector-set! ar i (vector-ref ar j))
              (vector-set! ar j temp)
              ;(display "swap done\n")
            )   ;end of swap
            ;(display ar)
          )
        )
        ;(display j)
        ;(display "\n")
        (set! j (+ j 1))
      )
```

```
      ;(display "\tout\n")
      (let ((temp 0))   ;swap ar[i+1] with ar[high]
          ;(display "temp is ")
          ;(display temp)
          ;(display "\n")
        (set! temp (vector-ref ar (+ i 1)))
          ;(display "temp is now ")
          ;(display temp)
          ;(display "\n")
        (vector-set! ar (+ i 1) (vector-ref ar high))
          ;(display "check\n")
        (vector-set! ar high temp)
        ;(display "swap done\n")
      )   ;end of swap
      ;(display ar)
      (+ i 1)
    )
    ;(display "partition ")
    ;(+ 1 1)
  )
)

(define quicksort
  (lambda (low high)
    ;(display low)
    ;(display " ")
    ;(display high)
    ;(display "\n")
    (cond ((not (integer? low)) (display "low is not a valid index\n"))
      ((not (integer? high)) (display "high is not a valid index\n"))
            ((not (and (> low -1) (> high -1))) (display "error one of these: low or high < 0\n"))
      ((< low high)
                  (let ((pi (partition low high)))
          (quicksort low (- pi 1))
          (quicksort (+ pi 1) high)
          ;(display "good ")
          ;(display pi)
          ;(display "\n")
        )
        ;(display "part\n")
      );all good
      ;(else
      ;  (display " done\n")
      ;)
    )

    ;(display ar)
    ;(display "\n")
    (vector-copy ar)
  )
)
(define main
  (lambda ()
    (display "unsorted:\n")
    (display ar)  ;global var instead of pass by reference
    (display "\nsorted:\n")
    (display (quicksort 0 (- size 1) ))
    (display "\n\nunsorted:\n")
    (set! ar ar2)
    (set! size size2)
    (display ar)
    (display "\nsorted:\n")
    (display (quicksort 0 (- size 1) ))
    (display "\n\nunsorted:\n")
    (set! ar ar3)
    (set! size size3)
    (display ar)
    (display "\nsorted:\n")
    (display (quicksort 0 (- size 1) ))
    (display "\n\nunsorted:\n")

  )
)

(main)
```

How easy/hard was it was to program?

I was not familiar with paradigm, therefore Scheme was hard to program. It almost like an entire new language for me.

The ease/difficulty of debugging:

Compare to programming the language, debugging isn't that hard as long as you learn it in a correct way.

The speed of execution:

Run time already shown in the screenshot above.