```
(define temp (+(*(/ 9 5) 100) 32)) temp

(define square (lambda (x) (*x x)))

(define isRightAngled
      (lambda (a b c)
            (= (+ (square a) (square b)) (square c))
      )
)

(define sum
      (lambda (n)
            (if (= n 1) 1
                  (+ n (sum (- n 1)))
            )
      )
)

(define fib
      (lambda (n)
            (if (= n 1) 1
                  (if (= n 2) 1
                        (+ (fib(- n 2)) (fib(- n 1)) )
                  )
            )
      )
)

(define listOfPairs
      (list (list 1 2) (list 3 4) (list 5 6) )
)

listOfPairs

(define extract4
      (lambda (lst)
            (if (null? lst)
                  lst
                  (if (= 4 (car (car lst) ) )
                        (car lst)
                        (if (= 4 (car (cdr (car lst) ) ) )
                              (car lst)
                              (extract4 (cdr lst) )
                        )
                  )
            )
      )
)
```

```
(define extract2
      (lambda (lst)
            (if (null? lst)
                  lst
                  (if (= 2 (car (car lst) ) )
                        (car lst)
                        (if (= 2 (car (cdr (car lst) ) ) )
                              (car lst)
                              (extract2 (cdr lst) )
                        )
                  )
            )
      )
)

(define add92
      (lambda (lst)
            (cons (list 9 2) lst )
      )
)

(define getSecondElement
      (lambda (lst)
            (if (null? lst) lst
            (cons (car (cdr (car lst) ) ) (getSecondElement(cdr lst) )
            )
      )
)

(define isOdd
      (lambda (n)
            (if (= n 1) #t
                  (if (= n 0)      #f
                        (isOdd(- n 2))
                  )
            )
      )
)

(define odd
      (lambda (lst)
            (if (null? lst) lst
                  (if (isOdd(car lst))
                        (cons (car lst) (odd(cdr lst)))
                        (odd(cdr lst))
                  )
            )
      )
)
```

```
(define length
      (lambda (lst)
            (if (null? lst) 0
                  (+ 1 (length (cdr lst)))
            )
      )
)

(define max
      (lambda (lst)
            (if (= (length lst) 1 )
                  (car lst)
                  (let ((x (max lst) ))
                        (if (> (car lst) x )
                              (car lst)
                              x
                        )
                  )
            )
      )
)
```