1. Binomial Function

```
#lang racket

(define (binomial N K)
(if (= K 0)
1
(if (= K N)
1
(+ (binomial (- N 1) K) (binomial (- N 1) (- K 1))
)
)
)
)

(binomial 4 0)
(binomial 8 8)
(binomial 3 2)
(binomial 7 4)
```

```
Welcome to DrRacket, version 8.8 [cs].
Language: Determine language from source; memory limit: 128 MB.
1
1
3
35
>
```

2. Modulus Function

```
#lang racket

(define (modulus N M)
(if (< N M)
N
(if (equal? N 0)
0
(modulus (- N M) M)
)
)
)

(modulus 9 5)
(modulus 7 9)
(modulus 100 37)
(modulus 20 5)
(modulus -11 3)
```

```
Welcome to DrRacket, version 8.8 [cs].
Language: racket, with debugging; memory limit: 128 MB.
4
7
26
0
-11
>
```

3. Binary to Decimal Function

```
#lang racket

(define (modulus N M)
(if (< N M)
N
(if (equal? N 0)
0
(modulus (- N M) M)
)
)
)

(define (binaryToDecimal binary count)
(if (equal? binary 0)
0
(+ (* (modulus binary 10) (expt 2 count))
(binaryToDecimal (quotient binary 10) (+ count 1))
)
)
)

(binaryToDecimal 0 0)
(binaryToDecimal 1011 0)
(binaryToDecimal 111111 0)
(binaryToDecimal 10001 0)
```

```
Welcome to DrRacket, version 8.8 [cs].
Language: racket, with debugging; memory limit: 128 MB.
0
11
63
17
>
```

4. Add Binary Function

```
#lang racket

(define (modulus N M)
(if (< N M)
N
(if (equal? N 0)
0
(modulus (- N M) M)
)
)
)

(define (binaryToDecimal binary count)
(if (equal? binary 0)
0
(+ (* (modulus binary 10) (expt 2 count))
(binaryToDecimal (quotient binary 10) (+ count 1))
)
)
)

(define (addBinaryNumbers binaryNumbersList)
(if (null? binaryNumbersList)
0
(+ (binaryToDecimal (car binaryNumbersList) 0)
(addBinaryNumbers (cdr binaryNumbersList))
)
)
)

(addBinaryNumbers '(1101 111 10 101))
(addBinaryNumbers '(0))
(addBinaryNumbers '(11011))
```

```
Welcome to DrRacket, version 8.8 [cs].
Language: racket, with debugging; memory limit: 128 MB.
27
0
27
>
```

5. Find Minimum Function

```
#lang racket

(define (findMinimum numbersList) (getMinimum (cdr numbersList) (car numbersList)))

(define (getMinimum numbersList minTillNow)
(if (null? numbersList)
minTillNow
(if (< (car numbersList) minTillNow)
(getMinimum (cdr numbersList) (car numbersList))
(getMinimum (cdr numbersList) minTillNow)
)
)
)

(findMinimum '(4 5 1 2 5))
(findMinimum '(3))
(findMinimum '(5 5 5))
(findMinimum '())
```

```
Welcome to DrRacket, version 8.8 [cs].
Language: racket, with debugging; memory limit: 128 MB.
1
3
5
cdr: contract violation
  expected: pair?
  given: '()
```

6. Remove Item Function

```
#lang racket

(define (removeAtom atom list)
(if (null? list)
list
(if (equal? atom (car list))
(removeAtom atom (cdr list))
(cons (car list) (removeAtom atom (cdr list)))
)
)
)

(removeAtom 'a '())
(removeAtom 'a '(a))
(removeAtom 'a '(a b c d a b a a))
(removeAtom 'a '(x y z))
(removeAtom 'a '(a (x y z) (r s t a)))
(removeAtom 'a '(((a (l a) b) a) m a))
```

```
Welcome to DrRacket, version 8.8 [cs].
Language: racket, with debugging; memory limit: 128 MB.
'()
'()
'(b c d b)
'(x y z)
'((x y z) (r s t a))
'(((a (l a) b) a) m)
>
```

7. Selection Sort Function

```
#lang racket

(define (findMinimum numbersList) (getMinimum (cdr numbersList) (car numbersList)))

(define (getMinimum numbersList minTillNow)
(if (null? numbersList)
minTillNow
(if (< (car numbersList) minTillNow)
(getMinimum (cdr numbersList) (car numbersList))
(getMinimum (cdr numbersList) minTillNow)
)
)
)

(define (removeAtom atom list)
(if (null? list)
list
(if (equal? atom (car list))
(cdr list)
(cons (car list) (removeAtom atom (cdr list)))
)
)
)

(define (selectionSort numbersList)
(if (null? numbersList)
'()
(cons (findMinimum numbersList)
(selectionSort (removeAtom (findMinimum numbersList) numbersList))
)
)
)

(selectionSort '())
(selectionSort '(5))
(selectionSort '(6 10 23 12 2 9 18 1 0 15))
(selectionSort '(3 4 7 3 7 7 4 3 2 3 7))
```

```
Welcome to DrRacket, version 8.8 [cs].
Language: racket, with debugging; memory limit: 128 MB.
'()
'(5)
'(0 1 2 6 9 10 12 15 18 23)
'(2 3 3 3 3 4 4 7 7 7 7)
>
```