#lang racket

;;-------General Functions------------

(define (map f L)

(if (null? L) '()

(cons (f (car L)) (map f (cdr L)))))

(define (isEven x)

(if (= (mode x 2) 0) #t #f))

(define (lstCount L v) (lambda (n)

;;cond, if = add 1, if not = add 0, if null exit

(if (= (cons x) v)

(+ n 1)

0)

(define (getDigit x)

(if (null? x) 0

(else modulo (quotient x 10)))

;;--------------A&B-------------------

;;mod 10 (n/10) to get current digit

;;if 9 replace with 0, else replace with add 1

;;recurse to go through each digit

;;cons results to list, list to number

(define (digitinc x)

(cond

((= (getDigit x) 9) 0)

(else (+ x 1) x)

(digitinc x))

;;--------------C---------------------

(define (listPicker L M)

;;--------------D---------------------

(define (neshlist L)

;;--------------E---------------------

(define (repChildren T K L R)

;;--------------F---------------------

;;map function, then count true and false returns

(define (functoinMajority F L)

(if (> (lstCount (map F L) #t) (lstCount (map F L) #f)) #t)

#f))

;;--------------G---------------------

;;

(define (makeMangle M)