1) (define (make5 n m)

(cond((= -2 (first2 m))-2)

((= -2 (last3 n))-2)

(else (+ (\* (last3 n) 100)(first2 m)))))

(define (first2 n)

(cond ((< n 0)(first2(- 0 n)))

((< n 10)-2)

((< n 100)n)

(else(first2 (floor(/ n 10))))))

(define (last3 n)

(cond ((< n 0)(last3 (- 0 n)))

((< n 100)-2)

(else (modulo n 1000))))

2) (define (concatL l1 l2)(cond ((null? (car l1))'())

((null? (cdr l1))(list(string-append (car l1) (car l2))))

(else (cons (string-append (car l1) (car l2)) (concatL (cdr l1) (cdr l2))))))

3) (define (buildListh N E)

(cond ((= N 1) (list E))

(else (cons E (buildListh (- N 1) E)))))

(define (buildList N E)(display(buildListh N E)))

4) (define (listpicket P L)

(cond ((null? P) ((null? L)'()))

((null? (cdr L)) (cons P(cons (car L) (list P))))

(else (cons P (cons (car L)(listpicket P (cdr L)))))))

5) (define (listpicketallh P L)

(cond

((empty? L) '())

((list? (car L))(cons(append (list P)(listpicketallh P (car L)))(append (list P)(listpicketallh P (cdr L)))))

(else (append (append (list(car L))(list P))(listpicketallh P (cdr L))))))

(define (listpicketall P L)(append (list P)(listpicketallh P L)))

6.)

(define (selectN n) (lambda(x)(select n x)))

(define (select n x)

(cond ((= n 0) x)

(else (select (- n 1)(cdr x)))))