Com S 342 HW4 Jinwoo Kim

Q1.

```
(define X 88)
(define Y 89)
(define Z 90)
(define adddef (lambda (n) (+ 32 n)))
(define DefineLang (lambda (letter)
                                (adddef letter)
                                    ))
```

Q2.

```
(define f (lambda (n)
   (if (= n 0)
         0
         (if (= n 1)
               1
               (+ (f (- n 1)) (f (- n 2)))))))
```

Q3.

Q3-(a).

```
(define count (lambda (n lst)
   (if (null? lst)
         0
         (if (= n (car lst))
               (+ 1 (count n (cdr lst)))
               (count n (cdr lst))))
))
```

Q3-(b)

```
(define max (lambda (lst)
                    (if (null? lst)
                        0
                        (if (null? (cdr lst))
                            (car lst)
                            (if (< (car lst) (car (cdr lst)))
                                (max (cdr lst))
                                (max (cons (car lst) (cdr (cdr lst))))
                                )
                            )
                        )
  )
```

Q3-(c)

```
(define repeat (lambda (lst)
                        (if (null? lst)
                            '()
                            (cons (count (car lst) (cdr lst)) (repeat (cdr lst)))
                            )
                        )
    )


(define maxrepeat(lambda (lst)
                            (if (null? lst)
                                0
                                (+ 1 (max (repeat lst)))
                                )))
```

Q4.

Q4-(a)

```
(define pairs
   (cons (cons 1 (list 5)) (cons (cons 6 (list 4)) (cons (cons 7 (list 8)) (list (cons 15 (list 10))))))
   )
```

Q4-(b)

```
(define (secondSum lst)
   (if (null? (cdr lst)) (car (cdr (car lst)))
         (+ (car (cdr (car lst))) (secondSum (cdr lst))))
```

Q5.

```
(define apair (pair 2 3))
```

Q5-(a)

```
(define second(lambda(p)(p #f)))
```

Q5-(b)

```
(define pair
   (lambda (fst snd)
      (lambda (op)
         (op fst snd))))
```

Q5-(c)

```
(define add
   (lambda (x y)
      (+ x y)))
(define FuncLang (lambda(p) (p add)))
```

Q6.

Q6-(a)

```
(define mylist (cons (cons 1 (list 3)) (cons (cons 4 (list 2)) (list (cons 5 (list 6))))))
```

Q6-(b)

```
(define add
   (lambda (x y)
      (+ x y)))
(define subtract
   (lambda (x y)
      (- x y)))
(define applyonnth
   (lambda (op lst n)
      (if (null? lst)
            -1
            (if (= n 1)
                  (op (car (car lst)) (car (cdr (car lst))))
                  (applyonnth op (cdr lst) (- n 1))
                  ))))
```

Q6-(c).

```
(define applyonnth
   (lambda (op)
       (lambda (n)
            (lambda (lst)
                (if (null? lst)
                      -1
                      (if (= n 1)
                            (op (car (car lst)) (car (cdr (car lst))))
                            (applyonnth op (cdr lst) (- n 1))
                            ))))))
```