Name: Wen Chuan Lee
Student ID: 4927941 (leex7095)
Class: CSCI 2041
Title: Homework 8: Lazy Evaluations - Solutions

---

**Question 1: Evaluate** `sum (take 3 (some_squares_from 5 1))`

Using Call By Value Semantics

```
sum (take 3 (some_squares_from 5 1)) - the initial expression
= sum (take 3 ((1*1) :: some_squares_from (5-1) (1+1)))
= sum (take 3 ((1*1) :: some_squares_from 4 (1+1)))
= sum (take 3 ((1*1) :: some_squares_from 4 2))
= sum (take 3 ((1*1) :: (2*2) :: some_squares_from (4-1) (2+1)))
= sum (take 3 ((1*1) :: (2*2) :: some_squares_from 3 (2+1)))
= sum (take 3 ((1*1) :: (2*2) :: some_squares_from 3 3))
= sum (take 3 ((1*1) :: (2*2) :: (3*3) :: some_squares_from (3-1) (3+1)))
= sum (take 3 ((1*1) :: (2*2) :: (3*3) :: some_squares_from 2 (3+1)))
= sum (take 3 ((1*1) :: (2*2) :: (3*3) :: some_squares_from 2 4))
= sum (take 3 ((1*1) :: (2*2) :: (3*3) :: (4*4) :: some_squares_from (2-1) 4+1))
= sum (take 3 ((1*1) :: (2*2) :: (3*3) :: (4*4) :: some_squares_from 1 (4+1)))
= sum (take 3 ((1*1) :: (2*2) :: (3*3) :: (4*4) :: some_squares_from 1 5))
= sum (take 3 ((1*1) :: (2*2) :: (3*3) :: (4*4) ::(5*5):: some_squares_from (1-1) 5+1))
= sum (take 3 ((1*1) :: (2*2) :: (3*3) :: (4*4) :: (5*5) :: some_squares_from 0 (5+1)))
= sum (take 3 ((1*1) :: (2*2) :: (3*3) :: (4*4) :: (5*5 :: some_squares_from 0 6))
= sum (take 3 ((1*1) :: (2*2) :: (3*3) :: (4*4) :: (5*5) :: [ ]))
= sum (take 3 ((1*1) :: (2*2) :: (3*3) :: (4*4) :: 25 :: [ ]))
= sum (take 3 ((1*1) :: (2*2) :: (3*3) :: 16 :: 25 :: [ ]))
= sum (take 3 ((1*1) :: (2*2) :: 9 :: 16 :: 25 :: [ ]))
= sum (take 3 ((1*1) :: 4 :: 9 :: 16 :: 25 :: [ ]))
= sum (take 3 (1 :: 4 :: 9 :: 16 :: 25 :: [ ]))
= sum (1 :: take (3-1) (4 :: 9 :: 16 :: 25 :: [ ]))
= sum (1 :: take 2 (4 :: 9 :: 16 :: 25 :: [ ]))
= sum (1 :: 4 :: take (2-1) (4 :: 9 :: 16 :: 25 :: [ ]))
= sum (1 :: 4 :: take 1 (4 :: 9 :: 16 :: 25 :: [ ]))
= sum (1 :: 4 :: 9 :: take 0 (9 :: 16 :: 25 :: [ ]))
= sum (1 :: 4 :: 9 :: [ ])
= 1 + (sum (4 :: 9 :: [ ]))
= 1 + (4 + (sum (9 :: [ ])))
= 1 + (4 + (9 + (sum [])))
= 1 + (4 + (9 + 0))
= 1 + (4 + 9)
= 1 + 13
= 14
```

## Using Call By Name

```
sum (take 3 (some_squares_from 5 1)) - expand to find a match
= sum (take 3 ((1*1) :: some_squares_from (5-1) (1+1))) - match found
= sum ((1*1) :: take (3-1) (some_squares_from (5-1) (1+1))
= (1*1) + sum (take (3-1) (some_squares_from (5-1) (1+1))
= 1 + (sum (take (3-1) (some_squares_from (5-1) (1+1))) - evaluated to match take expr
= 1 + (sum (take (3-1) (some_squares_from 4 (1+1)))
= 1 + (sum (take (3-1) ((1+1)*(1+1) :: some_squares_from (4-1) ((1+1)+1))))
   -  we need to check the 2nd take pattern and thus evaluate the first argument
= 1 + (sum (take 2 ((1+1)*(1+1) :: some_squares_from (4-1) ((1+1)+1))))
= 1 + (sum ((1+1)*(1+1) :: take (2-1) (some_squares_from (4-1) ((1+1)+1))))
= 1 + ((1+1)*(1+1) + sum (take (2-1) (some_squares_from (4-1) ((1+1)+1))))
= 1 + ((1+1)*2 + sum (take (2-1) (some_squares_from (4-1) ((1+1)+1))))
= 1 + (2*2 + sum (take (2-1) (some_squares_from (4-1) ((1+1)+1))))
= 1 + (4 + (sum (take (2-1) (some_squares_from (4-1) ((1+1)+1)))))
= 1 + (4 + (sum (take (2-1) (some_squares_from 3 ((1+1)+1)))))
= 1 + (4 + (sum (take (2-1)
                    (((1+1)+1))*((1+1)+1)) ::(some_squares_from (3-1) (((1+1)+1)+1))))
= 1 + (4 + (sum (take 1 (((1+1)+1))*((1+1)+1))
                    ::(some_squares_from (3-1) (((1+1)+1)+1))))
= 1 + (4 + (sum (((1+1)+1))*((1+1)+1)) ::
                    take (1-1) (some_squares_from (3-1) (((1+1)+1)+1))))
= 1 + (4 + (((1+1)+1))*((1+1)+1))
          +  sum (take (1-1) (some_squares_from (3-1) (((1+1)+1)+1)))
= 1 + (4 + (((1+1)+1))*((1+1)+1))
             + sum (take (1-1) (some_squares_from (3-1) (((1+1)+1)+1))))
= 1 + (4 + ((((2+1))*((1+1)+1))
             + sum (take (1-1) (some_squares_from (3-1) (((1+1)+1)+1))))
= 1 + (4 + ((3*((1+1)+1)) + sum (take (1-1) (some_squares_from (3-1) (((1+1)+1)+1))))
= 1 + (4 + ((3*(2+1)) + sum (take (1-1) (some_squares_from (3-1) (((1+1)+1)+1))))
= 1 + (4 + (((3*3) + sum (take (1-1) (some_squares_from (3-1) (((1+1)+1)+1))))
= 1 + (4 + (9 + sum (take (1-1) (some_squares_from (3-1) (((1+1)+1)+1)))))
= 1 + (4 + (9 + sum (take 0 (some_squares_from (3-1) (((1+1)+1)+1)))))
= 1 + (4 + (9 + sum (take 0 (some_squares_from 2 (((1+1)+1)+1)))))
= 1 + (4 + (9 + sum (
      take 0 ((((1+1)+1)+1)*(((1+1)+1)+1) :: some_squares_from (2-1) (((1+1)+1)+1))))
= 1 + (4 + (9 + (sum [])))
= 1 + (4 + (9 + 0))
= 1 + (4 + 9)
= 1 + 13
= 14
```

## Using Lazy Evaluation

```
sum (take 3 (some_squares_from 5 1))
= sum (take 3 ((1*1) :: some_squares_from (5-1) (1+1)))
= sum (v :: take (3-1) (some_squares_from (5-1) (1+1)) -where v = 1*1
= v + sum (take (3-1) (some_squares_from (5-1) (1+1)) -where v= 1*1
= 1 + sum (take (3-1) (some_squares_from (5-1) (1+1))
= 1 + sum (take (3-1) (some_squares_from 4 (1+1))
= 1 + sum (take (3-1) (v*v :: some_squares_from (4-1) v+1)) - where v = 1+1
= 1 + sum (take 2 (v*v :: some_squares_from (4-1) v+1)) - where v = 1+1)
= 1 + sum (v*v :: take (2-1) (some_squares_from (4-1) v+1)) - where v = 1+1
= 1 + (v*v + (sum (take (2-1) (some_squares_from (4-1) v+1)))) - where v = 1+1
= 1 + (4 + (sum (take (2-1) (some_squares_from (4-1) (2+1)))))
= 1 + (4 + (sum (take (2-1) (some_squares_from 3 (2+1)))))
= 1 + (4 + (sum (take 1 (some_squares_from 3 (2+1)))))
= 1 + (4 + (sum (take 1 (v*v :: some_squares_from (3-1) v+1)))) - where v=2+1
= 1 + (4 + (sum (v*v :: take (1-1) some_squares_from (3-1) v+1))) - where v=2+1
= 1 + (4 + (v*v + (sum (take (1-1) some_squares_from (3-1) v+1)))) - where v=2+1
= 1 + (4 + (9 + (sum (take (1-1) some_squares_from (3-1) 3+1))))
= 1 + (4 + (9 + (sum (take (1-1) some_squares_from 2 3+1))))
= 1 + (4 + (9 + (sum (take (1-1) (v*v :: some_squares_from (2-1) v+1)))) -where v = 3+1
= 1 + (4 + (9 + (sum (take 0 (v*v :: some_squares_from (2-1) v+1)))) -where v = 3+1
= 1 + (4 + (9 + (sum [])))
= 1 + (4 + (9 + 0))
= 1 + (4 + 9)
= 1 + 13
= 14
```

Given Functions:

```
sum [] = 0
sum x::xs -> x + sum xs

take 0 lst = [ ]
take n [ ] = [ ]
take n (x::xs) = x::take (n-1) xs

some_squares_from 0 v = [ ]
some_squares_from n v = v*v :: some_squares_from (n-1) (v+1)
```