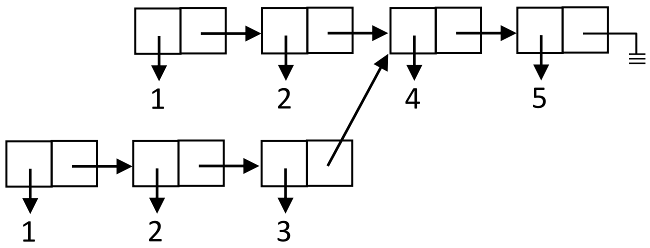
HW2 solution

1 a)




b) From part a), we see that if an element is inserted immediately after the $k^{\text{th}}$ element of a list, all the first $k$ elements of the list have to be copied. It follows that, for both time and space complexities, the worst-case insertion occurs when the element is going to be inserted at the end of the list.

Therefore, the worst-case input of the insertion sort is a list whose elements are in decreasing order.

For example,

$(n\; n-1 \cdots 3\; 2\; 1)$

is a worst-case input.

For the worst-case space complexity, let

$s(n)$ = # of cons cells allocated by function isort in the worst case on sorting a list of $n$ elements

then

$s(0) = 0$

$s(n) = s(n-1) + n - 1$

Clearly, $s(n) = O(n^2)$

So is the worst-case time complexity.

2~5 See file hw2sol.ss

6 a) Let Y = λf.(λx.f(xx))(λx.f(xx)) and G = λy.λf.f(yf)

Then,

Y G f

= (λx.G(xx)) (λx.G(xx)) f (1)

= G ((λx.G(xx)) (λx.G(xx))) f (2)

= f ((λx.G(xx)) (λx.G(xx))) f)

= f (Y G f) by (1)

Thus, YG is a fixed-point combinator.

Moreover, this fixed-point combinator can only work with lazy evaluation, because in step (2) the argument

(λx.G(xx)) (λx.G(xx))

passed to G has to be delayed.

b) Let Y = λf.(λx.f(λy.xxy))(λx.f(λy.xxy)) and G = λy.λf.f(λz.yfz)

Then,

Y G f

= (λx.G(λy.xxy)) (λx.G(λy.xxy)) f (1)

= G (λy.(λx.G(λy.xxy)) (λx.G(λy.xxy)) y) f (2)

= f (λz.(λy.(λx.G(λy.xxy)) (λx.G(λy.xxy)) y) f z) (3)

= f ((λy.(λx.G(λy.xxy)) (λx.G(λy.xxy)) y) f) (4)

= f ((λx.G(λy.xxy)) (λx.G(λy.xxy)) f)

= f (Y G f) by (1)

Thus, YG is a fixed point combinator.

Moreover, this combinator works with eager evaluation, because in step (2) the λ-exp (λy.(λx.G(λy.xxy)) (λx.G(λy.xxy)) y) passed to G in effect delays the argument (λx.G(xx)) (λx.G(xx)) mentioned in part b).

Note

(λy.(λx.G(λy.xxy)) (λx.G(λy.xxy)) y)

= (λx.G(λy.xxy)) (λx.G(λy.xxy) (5)

= (λx.G(xx)) (λx.G(xx)) (6)

Comment

Steps (3) to (6) make use of the so-called eta-conversion that says that, for any function f, f = λx.f x

c) See file hw2sol.ss

7 a) compile λx.λy.yx

= abstract x (compile λy.yx)

= abstract x (abstract y (compile yx))

= abstract x (abstract y yx)

= abstract x (S (abstract y y) (abstract y x))

= abstract x (S I (K x))

= S (abstract x (S I)) (abstract x (K x))

= S (S (abstract x S)) (abstract x I)) (abstract x (K x))

= S (S (K S) (K I)) (abstract x (K x))

= S (S (K S) (K I)) (S (abstract x K) (abstract x x)))

= S (S (K S) (K I)) (S (K K) I)

b) S (S (K S) (K I)) (S (K K) I) 2 + 3
= S (K S) (K I) 2 (S (K K) I 2) + 3
= K S 2 (K I 2) (S (K K) I 2) + 3
= S (K I 2) (S (K K) I 2) + 3
= K I 2 + (S (K K) I 2 +) 3
= I + (S (K K) I 2 +) 3
= + (S (K K) I 2 +) 3 reduce the argument of +
= + (K K 2 (I 2) +) 3
= + (K (I 2) +) 3
= + (I 2) 3
= + 2 3
= 5

c) See file hw2sol.ss