Implementación con listas:

mapS:

Implementación:

```
mapS  :: (a -> b) -> s a -> s b
mapS f []       = []
mapS f (x:xs) = let (rx,rxs) = (f x) ||| (mapS f xs)
                in rx:rxs
```

Trabajo:

$W_{mapS}(f, 0) = c_1$

$W_{mapS}(f, n) = k + W(f, s_0) + W_{mapS}(f, n-1) + k_1$

$= k_2 + W(f, s_0) + W_{mapS}(f, n-1)$

Luego de n iteraciones, vemos que la expresión del trabajo será de la forma:

$W_{mapS}(f, n) = \sum_{i=0}^{n-1} k_2 + \sum_{i=0}^{n-1} W(f, s_i)$

La sumatoria de las constantes queda dominada por la suma de las aplicaciones de f para cualquier complejidad de trabajo, por lo que se podría decir que:

$W_{mapS}(f, n) \leq k_3 \sum_{i=0}^{n-1} W(f, s_i)$

Por lo tanto, $W_{mapS}(f, n) \in O(\sum_{i=0}^{n-1} W(f, s_i))$.

Profundidad:

$S_{mapS}(f, 0) = c_1$

$S_{mapS}(f, n) = k + máx\ (S(f, s_0), S_{mapS}(f, n-1)) + k_1$

$= k_2 + máx\ (S(f, s_0), S_{mapS}(f, n-1))$

$= k_2 + máx\ (S(f, s_0), k_2 + máx\ (S(f, s_1), S_{mapS}(f, n-2)))$ *[1]*

Sea ahora $S_f = máx_{i=0}^{n-1} S(f, s_i)$. Podemos ver que $S_{mapS}(f, n)$ es menor o igual a la expresión [1] pero reemplazando todas las $S_f(s_i)$ por S_f, pues cada una de ellas es menor o igual a S_f. Por lo tanto, tenemos que:

$S_{mapS}(f, n) \leq k_2 + máx\ (S_f, k_2 + máx\ (S_f, k_2 + máx\ (... (k_2 + máx\ (S_f, c_1)\ )\ ...\ )))$.

Luego, suponiendo la S_f mayor que c_1, tendremos que el resultado de resolver todos los máx será:

$S_{mapS}(f, n) \leq (n-1)k_2 + S_f$

Entonces, $S_{mapS}(f, n) \in O(n + máx_{i=0}^{n-1} S(f, s_i))$.

appendS:

Implementación:

```
appendS :: s a -> s a -> s a
appendS = (++)
```

La función appendS fue definida utilizando la función del prelude (++). La siguiente está definida como:

```
(++) :: [a] -> [a] -> [a]
{-# NOINLINE [1] (++) #-}    -- We want the RULE to fire first.
                             -- It's recursive, so won't inline anyway,
                             -- but saying so is more explicit
(++) []     ys = ys
(++) (x:xs) ys = x : xs ++ ys
```

Por lo tanto, analizaremos los costos de trabajo y profundidad de dicha función para definir los costos de appendS. Notar que el costo de la función está asociado directamente al largo del primer argumento, por lo que omitiremos el segundo para el análisis.

Trabajo:

$W_{++}(0) = c_1$

$W_{++}(n) = k + W_{++}(n-1)$

Luego de n iteraciones vemos que el costo resulta lineal:

$W_{++}(n) = \sum_{i=0}^{n-1} k = k.n$

Por lo tanto, $W_{++}(n) \in O(n)$ donde n es el largo del primer argumento.

Finalmente, $W_{appendS}(n) = W_{++}(n) \in O(n)$

Profundidad:

$S_{++}(0) = c_1$

$S_{++}(n) = k + S_{++}(n-1)$

Luego de n iteraciones vemos que el costo resulta lineal:

$S_{++}(n) = \sum_{i=0}^{n-1} k = k.n$

Por lo tanto, $S_{++}(n) \in O(n)$ donde n es el largo del primer argumento.

Finalmente, $S_{appendS}(n) = S_{++}(n) \in O(n)$

reduceS

Implementación:

```
contraerS :: (a -> a -> a) -> [a] -> [a]
contraerS f []       = []
contraerS f [x]      = [x]
contraerS f (x:y:xs) = let (xyC,xsC) = (f x y) ||| (contraerS f xs)
                       in xyC : xsC

reduceS :: (a -> a -> a) -> a -> s a -> a
reduceS f base []  = base
reduceS f base [x] = f base x
reduceS f base xs  = reduceS f base (contraerS f xs)
```

Trabajo:

$W_{reduceS}(f, base, 0) = c_1$

$W_{reduceS}(f, base, 1) = c_2 + W(f, base, x)$

$W_{reduceS}(f, base, n) = k + W_{contraerS}(f, n) + W_{reduceS}(f, base, \lceil \frac{n}{2} \rceil)$ *[1]*

Para continuar resolviendo la expresión, se debe conocer el costo de contraerS.

$W_{contraerS}(f, 0) = c_1$

$W_{contraerS}(f, 1) = c_2$

$W_{contraerS}(f, n) = k + W(f, s_0, s_1) + W_{contraerS}(n-2)$

$$= \sum_{i=0}^{\lfloor (n-1)/2 \rfloor} k + \sum_{i=0}^{\lfloor (n-1)/2 \rfloor} W(f, s_{2i}, s_{2i+1})$$

De lo que podemos ver que $W_{contraerS}(f, n) \in O(\sum_{i=0}^{\lfloor (n-1)/2 \rfloor} W_f(s_i, s_{i+1}))$.

Por lo que podemos reemplazar en [1]:

$W_{reduceS}(f, base, n) \leq k + k_1 \sum_{i=0}^{\lfloor (n-1)/2 \rfloor} W_f(s_i, s_{i+1}) + W_{reduceS}(f, base, \frac{n}{2})$ *[2]*

Nótese que utilizamos suavidad para olvidarnos de la función techo en $W_{reduceS}$.

Si se continuara ampliando la expresión, aparecería ahora una sumatoria que dependería de las aplicaciones anteriores de f, lo cual complicaría expresar y desarrollar el cálculo de la recurrencia. Para calcular el trabajo de reduceS por fuera de las aplicaciones de la función argumento, se asume el trabajo de f constante. Retomando en [2] con esa consideración:

$W_{reduceS}(f, base, n) \leq k + k_2 n + W_{reduceS}(f, base, \frac{n}{2}) \leq k_3 n + W_{reduceS}(f, base, \frac{n}{2})$

Como ahora los casos base de la recurrencia son de trabajo constante, se puede utilizar el Teorema Maestro (tercer apartado) para llegar a que $W_{reduceS}(f, base, n) \in O(n)$.

Profundidad:

El razonamiento para el cálculo de la profundidad será muy similar al del trabajo. Primero se expresa la recurrencia para reduceS:

$S_{reduceS}(f, base, 0) = c_1$

$S_{reduceS}(f, base, 1) = c_2 + S(f, base, x)$
$S_{reduceS}(f, base, n) = k + S_{contraerS}(f, base, n) + S_{reduceS}(f, base, \lceil \frac{n}{2} \rceil)$ *[3]*

Luego se busca la profundidad de contraerS:
$S_{contraerS}(f, 0) = c_1$
$S_{contraerS}(f, 1) = c_2$
$S_{contraerS}(f, n) = k + máx(S_f(s_0, s_1) + S_{contraerS}(f, n-2))$

La cual es una expresión similar a la que se llegó en el desarrollo de la recurrencia de la profundidad de mapS. Razonando de manera análoga, se llega a que
$S_{contraerS}(f, n) \in O(n + máx_{i=0}^{\lfloor (n-1)/2 \rfloor} S(f, s_{2i}, s_{2i+1}))$.

Reemplazando con esa expresión en [3], se tiene que:
$S_{reduceS}(f, base, n) \leq k + k_1 n + k_1 máx_{i=0}^{\lfloor (n-1)/2 \rfloor} S(f, s_{2i}, s_{2i+1}) + S_{reduceS}(f, base, \frac{n}{2})$ *[4]*

Nótese que utilizamos suavidad para olvidarnos de la función techo en $S_{reduceS}$.
Ahora, se considera constante la profundidad de f a fines de calcular la profundidad de la reducción. De esta forma, reemplazando en [4]:

$$\begin{aligned} S_{reduceS}(f, base, n) &\leq k + k_1 n + k_1 k_2 + S_{reduceS}(f, base, \tfrac{n}{2}) \\ &= k_3 + k_1 n + S_{reduceS}(f, base, \tfrac{n}{2}) \\ &\leq k_4 n + S_{reduceS}(f, base, \tfrac{n}{2}) \end{aligned}$$

Similarmente al caso del trabajo, se utiliza el Teorema Maestro (tercer enunciado) para llegar a que $S_{reduceS}(f, base, n) \in O(n)$

scanS

Implementación:

```
expandirS :: (a -> a -> a) -> [a] -> [a] -> Int -> [a]
expandirS f [] [] indx                    = []
expandirS f [x] [x'] indx                 = [x']
expandirS f s@(x:y:xs) s'@(x':xs') indx = if (mod indx 2) == 0
                                  then x' : (expandirS f s s' (indx+1))
                                  else let (head,tail) = (f x' x) ||| (expandirS f xs xs' (indx+1))
                                       in head : tail

scanS  :: (a -> a -> a) -> a -> s a -> (s a, a)
scanS f base []  = (singletonS base, base)
scanS f base [x] = (singletonS base, f base x)
scanS f base xs  = let xsC = contraerS f xs
                       (xsSeq,xsRed) = scanS f base xsC
                   in (expandirS f xs xsSeq 0, xsRed)
```

Trabajo:

$W_{scanS}(f, base, 0) = c_1$

$W_{scanS}(f, base, 1) = c_2 + W(f, base, x)$

$W_{scanS}(f, base, n) = k + W_{contraerS}(f, n) + W_{scanS}(f, base, \lceil \frac{n}{2} \rceil) + W_{expandirS}(f, n, \lceil \frac{n}{2} \rceil, 0)$ *[5]*

Las cotas para el trabajo y profundidad de contraerS ya se calcularon en el apartado de reduceS, por lo que falta despejar las de expandirS. Utilizando suavidad, omitiremos el uso de la función techo. Para los casos base será:

$W_{expandirS}(f, 0, 0, i) = c_1$

$W_{expandirS}(f, 1, 1, i) = c_2$

Luego, en el caso recursivo, el if determina cómo sería la expresión de la recurrencia. Suponiendo que la condición es verdadera, quedaría:

$W_{expandirS}(f, n, \frac{n}{2}, i) = k_1 + W_{expandirS}(f, n, \frac{n}{2}, i+1)$

Mientras que si fuera falsa, la expresión sería:

$W_{expandirS}(f, n, \frac{n}{2}, i) = k_2 + W(f, s_0, s'_0) + W_{expandirS}(f, n-2, \frac{n}{2}-1, i+1)$

Como el cuarto argumento se incrementa en cada llamada recursiva y la condición del if depende de la paridad del mismo, combinando ambas expresiones y suponiendo que se parte de una condición verdadera (lo cual ocurre pues expandirS se llama desde scanS con el cuarto argumento valiendo 0), se tiene que:

$$
\begin{aligned}
W_{expandirS}(f, n, \tfrac{n}{2}, i) &= k_1 + W_{expandirS}(f, n, \tfrac{n}{2}, i+1) \\
&= k_1 + k_2 + W(f, s_0, s'_0) + W_{expandirS}(f, n-2, \tfrac{n}{2}-1, i+2) \\
&= k + W(f, s_0, s'_0) + W_{expandirS}(f, n-2, \tfrac{n}{2}-1, i+2) \\
&= \sum_{i=0}^{\lfloor (n-1)/2 \rfloor} k + \sum_{i=0}^{\lfloor (n-1)/2 \rfloor} W(f, s_{2i}, s'_i)
\end{aligned}
$$

Por lo que se tiene que $W_{expandirS}(f, n, \frac{n}{2}, 0) \in O(\sum_{i=0}^{\lfloor (n-1)/2 \rfloor} W(f, s_{2i}, s'_i))$.

Entonces, reemplazando en [5] con los resultados obtenidos:

$W_{scanS}(f, base, n) \leq k + k_1 \sum_{i=0}^{\lfloor (n-1)/2 \rfloor} W(f, s_i, s_{i+1}) + k_2 \sum_{i=0}^{\lfloor (n-1)/2 \rfloor} W(f, s_{2i}, s'_i) + W_{scanS}(f, base, \frac{n}{2})$ [6]

Nótese que utilizamos suavidad para olvidarnos de la función techo en W_{scanS}.
De manera similar a reduceS, si se quisiera ampliar la expresión anterior, se encontrarían aplicaciones de f cuyos argumentos dependen de los resultados de aplicaciones en la llamada recursiva anterior. Por lo tanto, se expresará la cota de la misma manera, asumiendo su costo constante.

Entonces, suponiendo que el trabajo de una aplicación de f es constante, la expresión [6] quedaría:

$W_{scanS}(f, base, n) \leq k + k_3 n + k_4 n + W_{scanS}(f, base, \frac{n}{2})$
$= k + k_5 n + W_{scanS}(f, base, \frac{n}{2})$
$\leq k_6 n + W_{scanS}(f, base, \frac{n}{2})$

Lo cual implicaría, aplicando el tercer enunciado del Teorema Maestro, que
$W_{scanS}(f, base, n) \in O(n)$

Profundidad:

Para la profundidad, se razonará de la misma forma que en reduceS:

$S_{scanS}(f, base, 0) = c_1$
$S_{scanS}(f, base, 1) = c_2 + S(f, base, x)$
$S_{scanS}(f, base, n) = k + S_{contraerS}(f, n) + S_{scanS}(f, base, \lceil \frac{n}{2} \rceil) + S_{expandirS}(f, n, \lceil \frac{n}{2} \rceil, 0)$ [7]

Siguiendo el razonamiento que se usó anteriormente para el cálculo del trabajo de expandirS, se llega a que:

$S_{expandirS}(f, 0, 0, i) = c_1$
$S_{expandirS}(f, 1, 1, i) = c_2$
$S_{expandirS}(f, n, \frac{n}{2}, i) = k_1 + S_{expandirS}(f, n, \frac{n}{2}, i+1)$
$= k_1 + k_2 + máx(S(f, s_0, s'_0), S_{expandirS}(f, n-2, \frac{n}{2}-1, i+2))$
$= k + máx(S(f, s_0, s'_0), S_{expandirS}(f, n-2, \frac{n}{2}-1, i+2))$

La cual es una expresión similar a la que se llegó en el desarrollo de la recurrencia de la profundidad de mapS. Razonando de manera análoga, se llega a que
$S_{expandirS}(f, n, \frac{n}{2}, 0) \in O(n + máx_{i=0}^{\lfloor (n-1)/2 \rfloor} S(f, s_{2i}, s'_i))$.

Reemplazando con esa expresión en [7]:

$S_{scanS}(f, base, n) \leq k + k_1 n + k_1 máx_{i=0}^{\lfloor (n-1)/2 \rfloor} S(f, s_{2i}, s_{2i+1}) +$
$k_2 n + k_2 máx_{i=0}^{\lfloor (n-1)/2 \rfloor} S(f, s_{2i}, s'_i) + S_{scanS}(f, base, \frac{n}{2})$
$= k + k_3 n + k_1 máx_{i=0}^{\lfloor (n-1)/2 \rfloor} S(f, s_{2i}, s_{2i+1}) +$

$$k_2 \max_{i=0}^{\lfloor (n-1)/2 \rfloor} S(f, s_{2i}, s'_i) + S_{scanS}(f, base, \tfrac{n}{2}) \qquad [8]$$

Nótese que utilizamos suavidad para olvidarnos de la función techo en S_{scanS}.
Ahora se considera constante la profundidad de f a fines de calcular la profundidad de la contracción y expansión. De esta forma, reemplazando en [8]:

$$\begin{aligned} S_{scanS}(f, base, n) &\leq k + k_3 n + k_1 k_4 + k_2 k_5 + S_{scanS}(f, base, \tfrac{n}{2}) \\ &= k_6 + k_3 n + S_{scanS}(f, base, \tfrac{n}{2}) \\ &\leq k_7 n + S_{scanS}(f, base, \tfrac{n}{2}) \end{aligned}$$

Lo cual implicaría, aplicando el tercer enunciado del Teorema Maestro, que
$S_{scanS}(f, base, n) \in O(n)$