Задача 3. *Задачата да се реши на един от езиците Scheme или Haskell. По-долу оградете името на езика, който сте избрали за вашето решение.*

Дадени са непразен списък от едноместни числови функции **`fl`** и непразен списък от числа **`xl`**. Казваме, че числото **`x`** е "неподвижна точка" на функцията **`f`**, ако **`f(x) = x`**. Да се попълнят по подходящ начин празните полета по-долу така, че за всички функции от **`fl`**, които имат неподвижна точка сред числата в **`xl`**, функцията **`sumMinFix`** да намира сумата на най-малките им такива неподвижни точки. Ако никоя функция от **`fl`** няма неподвижна точка сред числата в **`xl`**, функцията **`sumMinFix`** да връща числото 0. Помощната функция **`addDefault`** служи да осигури, че ако подаденият ѝ списък е празен, то в него се добавя една стойност по подразбиране.
Упътване: можете да използвате наготово функциите `apply`, `filter`, `foldr`, `map`, `min`, `minimum`, както и всички стандартни функции в R^{5}RS за Scheme и Prelude за Haskell.

Scheme

```
(define (addDefault val l)
  (if (null? l) (list val) l))

(define (sumMinFix fl xl)
  (_______________
    (_______________
      (lambda (f)
        (apply _______________
          (addDefault _______________
            (_______________
              (lambda (x) _____________________________ ) xl)))) fl)))
```

Пример:

```
(sumMinFix (list (lambda (x) (/ 1 x)) exp (lambda (x) (- (* 2 x) 3))))
           '(-2 -1 1 3)) → 2  (= -1 + 3)
```

Haskell

```
addDefault val [] = [val]
addDefault val l  = l

sumMinFix fl xl =
  _______________
     (_______________
        (\f -> _______________
                (addDefault _______________
                   [ _______________ | x <- xl, _____________________________ ])) fl)
```

Пример:

```
sumMinFix [ (1/), exp, \x -> 2*x - 3 ] [-2, -1, 1, 3] → 2   (= -1 + 3)
```