Задача 5. (15 т.) Списъкът $l_1 = (u_0..u_k)$ наричаме *подсписък* на списъка $l_2 = (v_0..v_n)$, ако $k \le n$ и съществува такова i, че $0 \le i \le n-k$ и $u_0 = v_i, u_1 = v_{i+1}, .., u_k = v_{i+k}$. Пример: списъкът (1 2) е подсписък на списъка (0 1 2 3), но не е подсписък на списъка (1 0 2).

Да се дефинира функция ***(count-sub l1 l2)*** на езика Scheme, която намира колко пъти списъкът от числа *l1* се среща като подсписък на списъка от числа *l2*.

Пример: *(count-sub ‘(1 1) ‘(1 1 1 2 1 1)) -> 3*

Задача 3. (10 т.) Нека са дадени следните изрази на езиците Haskell и Scheme. Моля, посочете каква е оценката на изразите на един от двата езика по ваш избор (попълнете едно от правоъгълните карета по-долу).

```
map (head [(\couple->fst couple + snd couple)])
    (foldr1 (++) [[(1,2)],[(3,4)]])
```

Оценка:_____

```
[zip [x] [x] | x <- [1..5]]
```

Оценка:_____

```
map (\(x:y:z)->x:z) [[1,2,3],[2,3,1],[3,1,2]]
```

Оценка:_____

```
(map
   (car (list (lambda (couple) (+ (car couple) (cdr couple)))))
   (apply append '( ( (1 . 2) ) ( (3 . 4) ) ) ) )
```

Оценка:_____

```
(map (lambda (x)
             (cons x (list x)))
    '(1 2 3 4 5))
```

Оценка:_____

```
(map (lambda (pred) (filter pred '(1 2 3 4 5)))
     (list even? odd?))
```

Оценка:_____

Задача 7. (10 т.) *Задачата да се реши на езика Scheme или Haskell. В началото на вашето решение посочете кой език сте избрали.*

А) Напишете функция `totalMin`, която за списък от едноместни числови функции връща тази функция f от списъка, за която $f(0)$ е минимално.

Б) Напишете функция `chainMinCompositions`, която получава като аргумент едноместна числова функция `f` и генерира безкрайния поток (за Хаскел – безкрайния списък) $F_0, F_1, F_2, \ldots$, където:

$F_0 = id$
$F_1 = f$
$F_i = F_{i-1} \circ F_{i-2}$, ако $i > 1$ и $F_{i-1}(j) \neq F_{i-2}(j)$, за някое цяло число $j \in [0, i]$
$F_i = totalMin\,\{F_0, F_1, \ldots, F_{i-1}\}$, ако $i > 1$ и $F_{i-1}(j) = F_{i-2}(j)$, за всяко цяло число $j \in [0, i]$

Забележка: с id е означена функцията „идентитет”, като id(x) = x за произволно x, а с $f \circ g$ е означена композицията на функциите на f и g, като $(f \circ g)(x) = f(g(x))$.

Задача 7. (10 т.) *Задачата да се реши на езика Scheme или Haskell. В началото на вашето решение посочете кой език сте избрали*.

Нека е даден списък L, който може да съдържа елементи от произволен тип. Напишете функция `permutations`, която получава такъв списък и връща списък с всички пермутации (възможни пренареждания) на неговите елементи. Резултатът да се върне като списък от списъци, в който всеки подсписък представя една пермутация на елементите на L.

Пример (Scheme):
`(permutations '(1 2 3))` → `( (1 2 3) (1 3 2) (2 1 3) (2 3 1) (3 1 2) (3 2 1) )`

Пример (Haskell):
`permutations [1,2,3]` → `[ [1,2,3], [1,3,2], [2,1,3], [2,3,1], [3,1,2], [3,2,1] ]`

Задача 3 (10 точки). Дадени са следните дефиниции на програмните езици Haskell и Scheme, от програмния код на които липсват части. Попълнете полетата, обозначени с _____ с необходимия програмен код така, че да се получат посочените желани оценки. Изберете само един от двата езика за решението на задачата и напишете името му в даденото за целта поле. Точки за задачата се дават само за избрания от вас език.

Избран език:

	Език Haskell:
1	**merge x [] = ____** **merge ___ ___ = ___** **merge (x:xs) (y:ys) = if x < y** **then _____** **else _____** израз: **merge [1,3,5,7] [2,2,6,10]** желана оценка: **[1,2,2,3,5,6,7,10]**
2	израз: **(____->[y|y<-_____,even ____])[1,2,3,4]** желана оценка: **[2,4]**
	Език Scheme:
1	**(define (merge l1 l2)** **(cond ((null? l1) ___)** **(_____)** **((< (car l1) (car l2)) _____)** **(else _____)))** израз: **(merge '(1 3 5 7) '(2 2 6 10))** желана оценка: **(1 2 2 3 5 6 7 10)**
2	израз: **((lambda (___) (filter _____)) '(1 2 3 4))** желана оценка: **(2,4)**

Задача 9. (3 т.) Като използвате единствено процедурите cons, car и cdr (и техните производни) чрез обръщения към l напишете израз, който:

1. конструира дадения списък, където l=(one two three):

а) (one (two three))

……

……

б) (one (two) three)

……

……

в) ((one two three))

……

……

2. има оценка стойността Harry, ако l има вида:

а) (Ann and Harry)

……

……

б) (((Ann)) and ((Harry)))

……

……

Задача 10. (4 т.) Оценете изразите:

```
((lambda (x y) (x y 8))(lambda (x y) (/ x y)) 2)
```

……

```
(accumulate + 1 (filter even? '(1 44 73 12 16 7)))
```

……

```
(let* ((x (list (lambda (x) (x 3 4))))
       (x (list (lambda (x) (+ x 5)) ((car x) +))))
       (cadr x))
```

……

Задача 3. Нека са дадени следните изрази на езиците Haskell и Scheme. Да се посочи каква е оценката на изразите. Изберете само един от двата езика за решението на задачата и напишете името му в даденото за целта поле. Точки за задачата се дават само за избрания от Вас език.

Избран език:

Haskell:	
1.	`[ x : [x] | x <- [ [1,2], [3,4] ] ]` Оценка:
2.	`[ map (f 5) [1,2,3] | f <- [ (+), (-), (*) ] ]` Оценка:
3.	`"a" : [ ['b', 'c'], "d" ]` Оценка:
Scheme:	
1.	`(map (lambda (x) (append (list x) x) ) '( (1 2) (3 4) ) )` Оценка:
2.	`(map (lambda (f) (map (lambda (x) (f 5 x)) '(1 2 3) ) ) (list + - *))` Оценка:
3.	`(apply list (list (quote +) (quote 5) 8) )` Оценка:

Задача 3.

1. Дадени да следните дефиниции на функция, съответно на програмните езици Haskell и Scheme, от програмния код на които липсват части. Да се попълнят полетата, обозначени с _____, с необходимия програмен код така, че да се получат посочените желани оценки.

2. Дадени са следните изрази, съответно на програмните езици Haskell и Scheme. Да се посочи каква е оценката на израза.

Изберете само един от двата езика за решението на задачата и напишете името му в даденото за целта поле. Точки за задачата се дават само за избрания от Вас език.

Избран език:

Haskell:	
1.	`filterByChar c ls = filter _____ ls` израз: `filterByChar 'o' ["cat", "cow", "dog"]` желана оценка: `["cow", "dog"]`
2.	`let (x:y):z = ["Curry"] in (x,y,z)` Оценка:
Scheme:	

1.	`(define (filterByElement x m)`
	`  (filter (lambda (__) _____) m))`
	
	израз: `(filterByElement 2 '( (1 2 3) (2 3 4) (3 4 5) ) )`
	желана оценка: `( (1 2 3) (2 3 4) )`

2.	`(apply + (map (lambda (l) (apply max l) ) '((5 -2) (1 9) (6 -8) )))` Оценка:

Задача 3. (12т.) Компресирано представяне на безкраен поток от числа наричаме такъв друг поток, за който неколкократното повторение на числото **0** е заменено от точкова двойка с първи елемент **0** и втори – броят на повторенията.
Например, ако началото на един поток е **(0 0 0 1 0 0 7 3 ...),** компресираното му представяне има начало **((0.3) 1 (0.2) 7 3 ...)**
Реализирайте двойка функции на Scheme, които по зададен безкраен поток, съдържащ някакви стойности, намират компресираното му представяне и обратно.

Задача 6. (12 точки) Да се дефинира на езика Scheme функция (generate-bin n), която генерира поток от всички естествени числа в интервала [n, +∞), представени чрез своя двоичен запис. Двоичният запис за едно число се представя като списък от нули и единици. Например:

- 1 → (1)
- 2 → (1 0)
- 10 → (1 0 1 0)

Примерни изпълнения:

(generate-bin 0) → Потокът е **(0) (1) (1 0) (1 1) (1 0 0) (1 0 1) …**
(generate-bin 10) → Потокът е **(1 0 1 0) (1 0 1 1) (1 1 0 0) (1 1 0 1) …**

Задача 6. (12 точки) Даден е списък L, който съдържа цифри (естествени числа, принадлежащи на интервала [0, 9]). Да се напише функция (find-max L), която намира най-голямото число, което може да се образува от цифрите в L. Ако L е празен, функцията find-max трябва да връща нула.

Пример:

(find-max '()) → 0
(find-max '(0 0 0)) → 0
(find-max '(1 1 9 8 9 3 4 6 7 0 0)) → 99876431100

Задача 7. (12 точки) Компресирано представяне на даден списък от стойности наричаме такъв списък от точкови двойки, за който неколкократното последователно срещане на един елемент е заменено от точкова двойка, първият елемент на която показва повторената стойност, а вторият – броя на повторенията. Например, некомпресираният списък **`(2 2 2 Stan 7 7 2)`** след компресия приема вида **`((2 . 3) (Stan . 1) (7 . 2) (2 . 1))`**. Да се реализират на езика Scheme двойка функции, които преобразуват некомпресиран списък в компресиран и обратно.

Задача 2 (5 т.) Да се дефинира на езика Хаскел функцията **`sumProd`**, която по даден списък от списъци от цели числа **`ll`** да намира сумата от произведенията на елементите на тези списъци в **`ll`**, които не съдържат отрицателни числа.

```
sumProd :: [[Integer]] -> Integer

sumProd ll = ....................................................
```

Задача 3 (15 т.) Дадени са следните дефиниции на типове на езика Хаскел, описващи:
Пътуване (**`Trip`**) като наредена тройка от крайна дестинация, дължина в километри и цена
Екскурзия (**`Tour`**) като списък от пътувания

```
type Trip = (String, Integer, Float)

type Tour = [Trip]
```

Да се дефинират функции:

- **`discount`**, която по дадена екскурзия намира нова екскурзия, в която цената на всяко пътуване с дължина над **`len`** км е намалена с 10%.

```
discount :: Tour -> Integer -> Tour

discount tour len = ..............................................
```

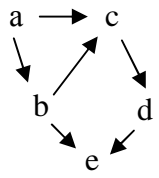
- **`shortenTour`**, която по дадена екскурзия намира нова екскурзия, в която всяка последователност от пътувания, започваща от дестинация **`from`** и завършваща в дестинация **`to`** е заменена с единственото пътуване **`trip`**. *Внимание*: последователност, отговаряща на описанието започва непосредствено след пътуване, чиято **крайна** дестинация е **`from`**, но не го включва!

```
shortenTour :: Tour -> String -> String -> Trip -> Tour

shortenTour tour from to trip = ....................................
```

Задача 7. (13 т.) Да се дефинира функция ***(ways g u v)*** на езика Scheme, намираща броя на различните пътища между върха ***u*** и върха ***v*** в ориентирания ацикличен граф ***g***, представен чрез асоциативен списък на наследниците.

Пример. Ако е даден графът



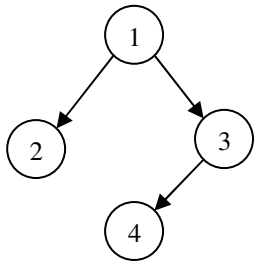
Представен чрез списъка *g = '((a b c) (b c e) (c d) (d e) (e))*, то оценката на *(ways g ‘a ‘e)* е 3.

Задача 8. (14 т.) Нека е дадено следното представяне на двоично дърво с произволни стойности по върховете:

- празният списък *()* е празно дърво
- ако t_1 и t_2 са две двоични дървета, то списакът с три елемента $(x\ t_1\ t_2)$ е двоично дърво със стойност на корена x, ляво поддърво t_1 и дясно поддърво t_2.

Да се дефинира функция *(leaves t)*, намираща списък от стойностите по листата на дървото t, представено по писания начин.

Пример: следното двоично дърво



Се представя чрез списъка *t = (1 (2 () ()) (3 (4 () ()) ()))*. За него *(leaves t) = (2 4)*.