Тема: **Обработка списков, множеств**  
  
I. Написать решение задачи на функциональном языке, проверить его в интерпретаторе HFL, вызвав функцию с разными аргументами.  
II. Переписать решение на Scheme и Kotlin и проверить аналогичным образом.

Определение функции для обработки списков на каждом из 3 языков программирования оценивается в 5 баллов. Оценка снижается за каждую ошибку на 1 балл.

В решениях на Kotlin функции определять как  
fun sqr(x:Double)=x\*x // функция, сокращенная форма  
а не

fun sqr(x:Double):Double { // функция, полная форма

return x\*x

}

Полная форма необходима только при процедурном (императивном) программировании, если нужны операторы for и while или для Unit-функций. Вместо операторов if и when в функциональном программировании нужно использовать if- и when-выражения (см. разницу в лекции на сайте). Оценка за определение функций на Kotlin с использованием return/for/while будет снижаться.

*Варианты заданий*

1. Заменить все вхождения положительных значений на 0 в списке чисел.

замена0('(1 2 -4 7)) → (0 0 -4 0)

2. Найти индекс k-го вхождения значения меньше 0 в списке чисел.

найти('(5 -2 -4 7),2) → 3

3. Количество четных значений в списке чисел.

колчет('(5 -2 6 7 10)) → 3

4. Удалить все отрицательные значения в списке чисел.

уд\_отр('(1 2 -4 7 -3)) → (1 2 7)

5. Максимальная длина непрерывной подпоследовательности из положительных значений в списке чисел.

максполчис('(1 2 -4 7 4 5 -3)) → 3

6. Длина первой непрерывной подпоследовательности из нечетных значений

в списке чисел.

длина\_неч('(0 2 3 -5 7 4 5 -3)) → 3

7. Заменить последнее вхождение положительного значения на 0

в списке чисел.

зам\_посл('(1 2 -4 7 4 5 -3)) → (1 2 -4 7 4 0 -3)

8. Заменить первое вхождение отрицательного значения

на максимальное значение в списке чисел.

зам\_макс('(1 2 -4 7 4 5 -3)) → (1 2 7 7 4 5 -3)

9. Вернуть список индексов отрицательных значений в списке чисел.

инд\_отр('(1 -2 -4 7 4 5 -3)) → (2 3 7)

10. Вернуть максимальное значение в списке чисел и

индекс его первого вхождения.

макс\_инд('(2 1 5 7 1 7 1)) → (7 4)

11. Подсписок с n-го элемента длиной m - подсписок(x,n,m)

подсписок('(2 1 5 7 1 7 1),3,4) → (5 7 1 7)

12. Найти минимум и количество минимальных значений

числового списка.

колмин('(2 1 5 7 1 5 1)) → (1 3)

13. Разбиение списка примерно пополам. Функция возвращает список из двух списков, в первом списке - элементы, стоящие на нечетных местах, во втором - на четных.

пополам('(A B C D E)) → ((A C E) (B D))

14. Пересечение множеств, заданных списками.

пересечение('(A B C D E),'(D F G A)) → (A D)

15. Функция, проверяющая, что числовой список является упорядоченным по возрастанию или убыванию.

упоряд('(1 3 4)) → #T

упоряд('(4 3 1 -2)) → #T

упоряд('(1 4 3)) → #F

16. Линеаризация списка

линеар('((A B) (C (D))) → (A B C D)

17. Заменить k-й элемент в списке чисел новым значением.

замена('(1 2 -4 7),2,5) → (1 5 -4 7)

18. Удалить n элементов из списка, начиная с k-го.

удчасть('(1 2 -4 3 7),2,3) → (1 7)

19. Вставить новый элемент перед k-м элементом в списке.

вставка('(1 5 -4 7),2,10) → (1 10 5 -4 7)

20. Удалить k-й элемент из списка.

удалить('(1 5 4 7),2) → (1 4 7)

21. Вставить новый подсписок перед k-м элементом в списке.

вставка('(1 5 -4 7),2,'(10 20)) → (1 10 20 5 -4 7)

22. Функция, проверяющая, что одно множество является подмножеством другого.

подмножество('(D B),'(A B C D E)) → #T

подмножество('(D B F),'(A B C D E)) → #F

23. Функция, проверящая, содержит ли список одинаковые элементы.

есть\_одинак('(A B C D B)) → #T

есть\_одинак('(B C D E)) → #F

24. Объединение множеств, заданных списками.

объединение('(A B C D E),'(D F G A)) → (A B C D E F G)

(порядок элементов в результате не важен)

25. Вставить число в упорядоченный числовой список.

вставка('(1 4 5 9),6)) → (1 4 5 6 9)