Лабораторная работа 1

Знакомство с языком Пролог. Синтаксис и основные типы данных

1. Загрузить среду программирования.

2. Выполнить задачи по варианту. Номер варианта равен номеру рабочего места.

3. Представить результат преподавателю.

Варианты:

1	1. Задайте функцию ball_volume(R), находящую объём шара радиуса R. Используйте библиотечную функцию math:pi(). ball_volume(1.0) => 4/3*pi (приближённо 4.18879...) 2. Задайте функцию from_to(N, M), строящую список целых чисел от N до M включительно. from_to(1, 4) => [1, 2, 3, 4] 3. Задайте функцию delta(List), которая заменяет каждый элемент списка, кроме первого, на его разность с предыдущим. delta([1,2,4,3]) => [1,1,2,-1] 4. Задайте функцию int_to_binary(N), которая возвращает двоичную запись целого числа N в виде строки (про строки см. lab1.txt) int_to_binary(8) => "1000" int_to_binary(-2) => "-10" 5. Задайте функцию rle_encode(List), которая кодирует список List методом повторов. Если подряд встречается несколько равных элементов, они заменяются на пару {Элемент, Число_повторов}; остальные элементы сохраняются. rle_encode([a,a,a,b,c,c,a,a]) => [{a,3},b,{c,2},{a,2}]
2	1. Задайте функцию seconds(Hours, Minutes, Seconds), вычисляющую, сколько секунд прошло с начала дня по заданному времени (даны число часов, минут и секунд). seconds(1, 2, 1) => 3721 (время 1:02:01) 2. Задайте функцию min(List), возвращающую минимальный элемент списка List. В случае пустого списка она должна выкидывать исключение. min([6,1,4]) => 1 3. Задайте функцию distinct(List), возвращающую true, если все элементы списка List различаются (и false, если нет). distinct([4,2,a,false]) => true distinct([1,2,2,3]) => false 4. Задайте функцию split_all(List, N), разбивающую список на части длиной N каждая (возможно, кроме последней). split_all([1, 2, 3, 4, 5], 3) => [[1, 2, 3], [4, 5]]

	5. Задайте функцию sublist(List, N, M), возвращающую отрезок списка List с N-ого по M-ый элемент (нумерация начинается с первого). sublist([1, 3, 4, 5, 6], [2, 4]) => [3, 4, 5]
3	1. Задайте функцию distance(P1, P2), находящую расстояние между точками P1 и P2, каждая из которых задана как кортеж из двух чисел. Используйте библиотечную функцию math:sqrt(). distance({1.0, 2.0}, {0.0, 1.0}) => sqrt(2) (приближённо 1.4142...) 2. Задайте функцию insert(List, X), которая получает отсортированный в порядке возрастания список List и число X, и добавляет X в List так, чтобы снова получить список в порядке возрастания. insert([1, 1.5, 2, 2.5, 3.5], 3) => [1, 1.5, 2, 2.5, 3, 3.5]. 3. Задайте функцию drop_every(List, N), удаляющую каждый N-ый элемент из списка List. drop_every([1,2,3,4,5,6,7], 2) => [1,3,5,7] drop_every([1,2,3,4,5,6,7], 3) => [1,2,4,5,7] 4. Задайте функцию rle_decode(EncodedList), которая работает противоположно функции rle_encode из варианта 1. decode([{a,3},b,{c,2},{a,2}]) => [a,a,a,b,c,c,a,a] 5. Задайте функцию diagonal(Matrix), которая возвращает диагональ матрицы, заданной как список списков. diagonal([[1,2,3], [4,5,6], [7,8,9]]) => [1,5,9]
4	1. Задайте функцию num_roots(A, B, C), находящую число корней квадратного уравнения A*x^2 + B*x + C = 0. num_roots(1, 0, -2) => 2 (т.к. уравнение 1*x^2 + 0*x - 2 = 0 имеет 2 корня) 2. Задайте функцию init(List), возвращающую список List без последнего элемента. init([1,2,3,4]) => [1,2,3] 3. Задайте функцию split(List, N), которая делит список List на две части: первые N элементов и всё, что идёт за ними. split([1, 3, 4, 5], 2) => {[1, 3], [4, 5]} 4. Задайте функцию binary_to_int(Bin), которая переводит двоичную запись числа (в виде строки Bin) в само это число. binary_to_int("100") => 4 binary_to_int("-101") => -5 5. Задайте функцию sliding_average(List, WindowSize), которая возвращает скользящее среднее списка List с размером окна WindowSize sliding_average([1, 2, 3, 4, 5, 6], 3) => [(1+2+3)/3, (2+3+4)/3, (3+4+5)/3, (4+5+6)/3] == [2.0, 3.0, 4.0, 5.0]
5	1. Задайте функцию ball_volume(R), находящую объём шара радиуса R. Используйте библиотечную функцию math:pi(). ball_volume(1.0) => 4/3*pi (приближённо 4.18879...) 2. Задайте функцию from_to(N, M), строящую список целых чисел от N до M включительно.

	from_to(1, 4) => [1, 2, 3, 4] 3. Задайте функцию delta(List), которая заменяет каждый элемент списка, кроме первого, на его разность с предыдущим. delta([1,2,4,3]) => [1,1,2,-1] 4. Задайте функцию int_to_binary(N), которая возвращает двоичную запись целого числа N в виде строки (про строки см. lab1.txt) int_to_binary(8) => "1000" int_to_binary(-2) => "-10" 5. Задайте функцию rle_encode(List), которая кодирует список List методом повторов. Если подряд встречается несколько равных элементов, они заменяются на пару {Элемент, Число_повторов}; остальные элементы сохраняются. rle_encode([a,a,a,b,c,c,a,a]) => [{a,3},b,{c,2},{a,2}]
6	1. Задайте функцию seconds(Hours, Minutes, Seconds), вычисляющую, сколько секунд прошло с начала дня по заданному времени (даны число часов, минут и секунд). seconds(1, 2, 1) => 3721 (время 1:02:01) 2. Задайте функцию min(List), возвращающую минимальный элемент списка List. В случае пустого списка она должна выкидывать исключение. min([6,1,4]) => 1 3. Задайте функцию distinct(List), возвращающую true, если все элементы списка List различаются (и false, если нет). distinct([4,2,a,false]) => true distinct([1,2,2,3]) => false 4. Задайте функцию split_all(List, N), разбивающую список на части длиной N каждая (возможно, кроме последней). split_all([1, 2, 3, 4, 5], 3) => [[1, 2, 3], [4, 5]] 5. Задайте функцию sublist(List, N, M), возвращающую отрезок списка List с N-ого по M-ый элемент (нумерация начинается с первого). sublist([1, 3, 4, 5, 6], [2, 4]) => [3, 4, 5]
7	1. Задайте функцию distance(P1, P2), находящую расстояние между точками P1 и P2, каждая из которых задана как кортеж из двух чисел. Используйте библиотечную функцию math:sqrt(). distance({1.0, 2.0}, {0.0, 1.0}) => sqrt(2) (приближённо 1.4142...) 2. Задайте функцию insert(List, X), которая получает отсортированный в порядке возрастания список List и число X, и добавляет X в List так, чтобы снова получить список в порядке возрастания. insert([1, 1.5, 2, 2.5, 3.5], 3) => [1, 1.5, 2, 2.5, 3, 3.5]. 3. Задайте функцию drop_every(List, N), удаляющую каждый N-ый элемент из списка List. drop_every([1,2,3,4,5,6,7], 2) => [1,3,5,7] drop_every([1,2,3,4,5,6,7], 3) => [1,2,4,5,7]

	4. Задайте функцию rle_decode(EncodedList), которая работает противоположно функции rle_encode из варианта 1. decode([{a,3},b,{c,2},{a,2}]) => [a,a,a,b,c,c,a,a] 5. Задайте функцию diagonal(Matrix), которая возвращает диагональ матрицы, заданной как список списков. diagonal([[1,2,3], [4,5,6], [7,8,9]]) => [1,5,9]
8	1. Задайте функцию num_roots(A, B, C), находящую число корней квадратного уравнения A*x^2 + B*x + C = 0. num_roots(1, 0, -2) => 2 (т.к. уравнение 1*x^2 + 0*x - 2 = 0 имеет 2 корня) 2. Задайте функцию init(List), возвращающую список List без последнего элемента. init([1,2,3,4]) => [1,2,3] 3. Задайте функцию split(List, N), которая делит список List на две части: первые N элементов и всё, что идёт за ними. split([1, 3, 4, 5], 2) => {[1, 3], [4, 5]} 4. Задайте функцию binary_to_int(Bin), которая переводит двоичную запись числа (в виде строки Bin) в само это число. binary_to_int("100") => 4 binary_to_int("-101") => -5 5. Задайте функцию sliding_average(List, WindowSize), которая возвращает скользящее среднее списка List с размером окна WindowSize sliding_average([1, 2, 3, 4, 5, 6], 3) => [(1+2+3)/3, (2+3+4)/3, (3+4+5)/3, (4+5+6)/3] == [2.0, 3.0, 4.0, 5.0]
9	1. Задайте функцию ball_volume(R), находящую объём шара радиуса R. Используйте библиотечную функцию math:pi(). ball_volume(1.0) => 4/3*pi (приближённо 4.18879...) 2. Задайте функцию from_to(N, M), строящую список целых чисел от N до M включительно. from_to(1, 4) => [1, 2, 3, 4] 3. Задайте функцию delta(List), которая заменяет каждый элемент списка, кроме первого, на его разность с предыдущим. delta([1,2,4,3]) => [1,1,2,-1] 4. Задайте функцию int_to_binary(N), которая возвращает двоичную запись целого числа N в виде строки (про строки см. lab1.txt) int_to_binary(8) => "1000" int_to_binary(-2) => "-10" 5. Задайте функцию rle_encode(List), которая кодирует список List методом повторов. Если подряд встречается несколько равных элементов, они заменяются на пару {Элемент, Число_повторов}; остальные элементы сохраняются. rle_encode([a,a,a,b,c,c,a,a]) => [{a,3},b,{c,2},{a,2}]
10	1. Задайте функцию seconds(Hours, Minutes, Seconds), вычисляющую, сколько секунд прошло с начала дня по заданному времени (даны число часов, минут и секунд).

	seconds(1, 2, 1) => 3721 (время 1:02:01) 2. Задайте функцию min(List), возвращающую минимальный элемент списка List. В случае пустого списка она должна выкидывать исключение. min([6,1,4]) => 1 3. Задайте функцию distinct(List), возвращающую true, если все элементы списка List различаются (и false, если нет). distinct([4,2,a,false]) => true distinct([1,2,2,3]) => false 4. Задайте функцию split_all(List, N), разбивающую список на части длиной N каждая (возможно, кроме последней). split_all([1, 2, 3, 4, 5], 3) => [[1, 2, 3], [4, 5]] 5. Задайте функцию sublist(List, N, M), возвращающую отрезок списка List с N-ого по M-ый элемент (нумерация начинается с первого). sublist([1, 3, 4, 5, 6], [2, 4]) => [3, 4, 5]
11	1. Задайте функцию distance(P1, P2), находящую расстояние между точками P1 и P2, каждая из которых задана как кортеж из двух чисел. Используйте библиотечную функцию math:sqrt(). distance({1.0, 2.0}, {0.0, 1.0}) => sqrt(2) (приближённо 1.4142...) 2. Задайте функцию insert(List, X), которая получает отсортированный в порядке возрастания список List и число X, и добавляет X в List так, чтобы снова получить список в порядке возрастания. insert([1, 1.5, 2, 2.5, 3.5], 3) => [1, 1.5, 2, 2.5, 3, 3.5]. 3. Задайте функцию drop_every(List, N), удаляющую каждый N-ый элемент из списка List. drop_every([1,2,3,4,5,6,7], 2) => [1,3,5,7] drop_every([1,2,3,4,5,6,7], 3) => [1,2,4,5,7] 4. Задайте функцию rle_decode(EncodedList), которая работает противоположно функции rle_encode из варианта 1. decode([{a,3},b,{c,2},{a,2}]) => [a,a,a,b,c,c,a,a] 5. Задайте функцию diagonal(Matrix), которая возвращает диагональ матрицы, заданной как список списков. diagonal([[1,2,3], [4,5,6], [7,8,9]]) => [1,5,9]
12	1. Задайте функцию num_roots(A, B, C), находящую число корней квадратного уравнения A*x^2 + B*x + C = 0. num_roots(1, 0, -2) => 2 (т.к. уравнение 1*x^2 + 0*x - 2 = 0 имеет 2 корня) 2. Задайте функцию init(List), возвращающую список List без последнего элемента. init([1,2,3,4]) => [1,2,3] 3. Задайте функцию split(List, N), которая делит список List на две части: первые N элементов и всё, что идёт за ними. split([1, 3, 4, 5], 2) => {[1, 3], [4, 5]}

	4. Задайте функцию binary_to_int(Bin), которая переводит двоичную запись числа (в виде строки Bin) в само это число. binary_to_int("100") => 4 binary_to_int("-101") => -5 5. Задайте функцию sliding_average(List, WindowSize), которая возвращает скользящее среднее списка List с размером окна WindowSize sliding_average([1, 2, 3, 4, 5, 6], 3) => [(1+2+3)/3, (2+3+4)/3, (3+4+5)/3, (4+5+6)/3] == [2.0, 3.0, 4.0, 5.0]
13	1. Задайте функцию ball_volume(R), находящую объём шара радиуса R. Используйте библиотечную функцию math:pi(). ball_volume(1.0) => 4/3*pi (приближённо 4.18879...) 2. Задайте функцию from_to(N, M), строящую список целых чисел от N до M включительно. from_to(1, 4) => [1, 2, 3, 4] 3. Задайте функцию delta(List), которая заменяет каждый элемент списка, кроме первого, на его разность с предыдущим. delta([1,2,4,3]) => [1,1,2,-1] 4. Задайте функцию int_to_binary(N), которая возвращает двоичную запись целого числа N в виде строки (про строки см. lab1.txt) int_to_binary(8) => "1000" int_to_binary(-2) => "-10" 5. Задайте функцию rle_encode(List), которая кодирует список List методом повторов. Если подряд встречается несколько равных элементов, они заменяются на пару {Элемент, Число_повторов}; остальные элементы сохраняются. rle_encode([a,a,a,b,c,c,a,a]) => [{a,3},b,{c,2},{a,2}]
14	1. Задайте функцию seconds(Hours, Minutes, Seconds), вычисляющую, сколько секунд прошло с начала дня по заданному времени (даны число часов, минут и секунд). seconds(1, 2, 1) => 3721 (время 1:02:01) 2. Задайте функцию min(List), возвращающую минимальный элемент списка List. В случае пустого списка она должна выкидывать исключение. min([6,1,4]) => 1 3. Задайте функцию distinct(List), возвращающую true, если все элементы списка List различаются (и false, если нет). distinct([4,2,a,false]) => true distinct([1,2,2,3]) => false 4. Задайте функцию split_all(List, N), разбивающую список на части длиной N каждая (возможно, кроме последней). split_all([1, 2, 3, 4, 5], 3) => [[1, 2, 3], [4, 5]] 5. Задайте функцию sublist(List, N, M), возвращающую отрезок списка List с N-ого по M-ый элемент (нумерация начинается с первого). sublist([1, 3, 4, 5, 6], [2, 4]) => [3, 4, 5]

15	1. Задайте функцию distance(P1, P2), находящую расстояние между точками P1 и P2, каждая из которых задана как кортеж из двух чисел. Используйте библиотечную функцию math:sqrt(). distance({1.0, 2.0}, {0.0, 1.0}) => sqrt(2) (приближённо 1.4142...) 2. Задайте функцию insert(List, X), которая получает отсортированный в порядке возрастания список List и число X, и добавляет X в List так, чтобы снова получить список в порядке возрастания. insert([1, 1.5, 2, 2.5, 3.5], 3) => [1, 1.5, 2, 2.5, 3, 3.5]. 3. Задайте функцию drop_every(List, N), удаляющую каждый N-ый элемент из списка List. drop_every([1,2,3,4,5,6,7], 2) => [1,3,5,7] drop_every([1,2,3,4,5,6,7], 3) => [1,2,4,5,7] 4. Задайте функцию rle_decode(EncodedList), которая работает противоположно функции rle_encode из варианта 1. decode([{a,3},b,{c,2},{a,2}]) => [a,a,a,b,c,c,a,a] 5. Задайте функцию diagonal(Matrix), которая возвращает диагональ матрицы, заданной как список списков. diagonal([[1,2,3], [4,5,6], [7,8,9]]) => [1,5,9]
16	1. Задайте функцию num_roots(A, B, C), находящую число корней квадратного уравнения A*x^2 + B*x + C = 0. num_roots(1, 0, -2) => 2 (т.к. уравнение 1*x^2 + 0*x - 2 = 0 имеет 2 корня) 2. Задайте функцию init(List), возвращающую список List без последнего элемента. init([1,2,3,4]) => [1,2,3] 3. Задайте функцию split(List, N), которая делит список List на две части: первые N элементов и всё, что идёт за ними. split([1, 3, 4, 5], 2) => {[1, 3], [4, 5]} 4. Задайте функцию binary_to_int(Bin), которая переводит двоичную запись числа (в виде строки Bin) в само это число. binary_to_int("100") => 4 binary_to_int("-101") => -5 5. Задайте функцию sliding_average(List, WindowSize), которая возвращает скользящее среднее списка List с размером окна WindowSize sliding_average([1, 2, 3, 4, 5, 6], 3) => [(1+2+3)/3, (2+3+4)/3, (3+4+5)/3, (4+5+6)/3] == [2.0, 3.0, 4.0, 5.0]
17	1. Задайте функцию ball_volume(R), находящую объём шара радиуса R. Используйте библиотечную функцию math:pi(). ball_volume(1.0) => 4/3*pi (приближённо 4.18879...) 2. Задайте функцию from_to(N, M), строящую список целых чисел от N до M включительно. from_to(1, 4) => [1, 2, 3, 4] 3. Задайте функцию delta(List), которая заменяет каждый элемент списка, кроме первого, на его разность с предыдущим.

	delta([1,2,4,3]) => [1,1,2,-1] 4. Задайте функцию int_to_binary(N), которая возвращает двоичную запись целого числа N в виде строки (про строки см. lab1.txt) int_to_binary(8) => "1000" int_to_binary(-2) => "-10" 5. Задайте функцию rle_encode(List), которая кодирует список List методом повторов. Если подряд встречается несколько равных элементов, они заменяются на пару {Элемент, Число_повторов}; остальные элементы сохраняются. rle_encode([a,a,a,b,c,c,a,a]) => [{a,3},b,{c,2},{a,2}]
18	1. Задайте функцию seconds(Hours, Minutes, Seconds), вычисляющую, сколько секунд прошло с начала дня по заданному времени (даны число часов, минут и секунд). seconds(1, 2, 1) => 3721 (время 1:02:01) 2. Задайте функцию min(List), возвращающую минимальный элемент списка List. В случае пустого списка она должна выкидывать исключение. min([6,1,4]) => 1 3. Задайте функцию distinct(List), возвращающую true, если все элементы списка List различаются (и false, если нет). distinct([4,2,a,false]) => true distinct([1,2,2,3]) => false 4. Задайте функцию split_all(List, N), разбивающую список на части длиной N каждая (возможно, кроме последней). split_all([1, 2, 3, 4, 5], 3) => [[1, 2, 3], [4, 5]] 5. Задайте функцию sublist(List, N, M), возвращающую отрезок списка List с N-ого по M-ый элемент (нумерация начинается с первого). sublist([1, 3, 4, 5, 6], [2, 4]) => [3, 4, 5]
19	1. Задайте функцию distance(P1, P2), находящую расстояние между точками P1 и P2, каждая из которых задана как кортеж из двух чисел. Используйте библиотечную функцию math:sqrt(). distance({1.0, 2.0}, {0.0, 1.0}) => sqrt(2) (приближённо 1.4142...) 2. Задайте функцию insert(List, X), которая получает отсортированный в порядке возрастания список List и число X, и добавляет X в List так, чтобы снова получить список в порядке возрастания. insert([1, 1.5, 2, 2.5, 3.5], 3) => [1, 1.5, 2, 2.5, 3, 3.5]. 3. Задайте функцию drop_every(List, N), удаляющую каждый N-ый элемент из списка List. drop_every([1,2,3,4,5,6,7], 2) => [1,3,5,7] drop_every([1,2,3,4,5,6,7], 3) => [1,2,4,5,7] 4. Задайте функцию rle_decode(EncodedList), которая работает противоположно функции rle_encode из варианта 1. decode([{a,3},b,{c,2},{a,2}]) => [a,a,a,b,c,c,a,a]

	5. Задайте функцию diagonal(Matrix), которая возвращает диагональ матрицы, заданной как список списков. diagonal([[1,2,3], [4,5,6], [7,8,9]]) => [1,5,9]
20	1. Задайте функцию num_roots(A, B, C), находящую число корней квадратного уравнения A*x^2 + B*x + C = 0. num_roots(1, 0, -2) => 2 (т.к. уравнение 1*x^2 + 0*x - 2 = 0 имеет 2 корня) 2. Задайте функцию init(List), возвращающую список List без последнего элемента. init([1,2,3,4]) => [1,2,3] 3. Задайте функцию split(List, N), которая делит список List на две части: первые N элементов и всё, что идёт за ними. split([1, 3, 4, 5], 2) => {[1, 3], [4, 5]} 4. Задайте функцию binary_to_int(Bin), которая переводит двоичную запись числа (в виде строки Bin) в само это число. binary_to_int("100") => 4 binary_to_int("-101") => -5 5. Задайте функцию sliding_average(List, WindowSize), которая возвращает скользящее среднее списка List с размером окна WindowSize sliding_average([1, 2, 3, 4, 5, 6], 3) => [(1+2+3)/3, (2+3+4)/3, (3+4+5)/3, (4+5+6)/3] == [2.0, 3.0, 4.0, 5.0]
21	1. Задайте функцию ball_volume(R), находящую объём шара радиуса R. Используйте библиотечную функцию math:pi(). ball_volume(1.0) => 4/3*pi (приближённо 4.18879...) 2. Задайте функцию from_to(N, M), строящую список целых чисел от N до M включительно. from_to(1, 4) => [1, 2, 3, 4] 3. Задайте функцию delta(List), которая заменяет каждый элемент списка, кроме первого, на его разность с предыдущим. delta([1,2,4,3]) => [1,1,2,-1] 4. Задайте функцию int_to_binary(N), которая возвращает двоичную запись целого числа N в виде строки (про строки см. lab1.txt) int_to_binary(8) => "1000" int_to_binary(-2) => "-10" 5. Задайте функцию rle_encode(List), которая кодирует список List методом повторов. Если подряд встречается несколько равных элементов, они заменяются на пару {Элемент, Число_повторов}; остальные элементы сохраняются. rle_encode([a,a,a,b,c,c,a,a]) => [{a,3},b,{c,2},{a,2}]
22	1. Задайте функцию seconds(Hours, Minutes, Seconds), вычисляющую, сколько секунд прошло с начала дня по заданному времени (даны число часов, минут и секунд). seconds(1, 2, 1) => 3721 (время 1:02:01) 2. Задайте функцию min(List), возвращающую минимальный элемент списка List. В случае пустого списка она должна выкидывать исключение.

	min([6,1,4]) => 1 3. Задайте функцию distinct(List), возвращающую true, если все элементы списка List различаются (и false, если нет). distinct([4,2,a,false]) => true distinct([1,2,2,3]) => false 4. Задайте функцию split_all(List, N), разбивающую список на части длиной N каждая (возможно, кроме последней). split_all([1, 2, 3, 4, 5], 3) => [[1, 2, 3], [4, 5]] 5. Задайте функцию sublist(List, N, M), возвращающую отрезок списка List с N-ого по M-ый элемент (нумерация начинается с первого). sublist([1, 3, 4, 5, 6], [2, 4]) => [3, 4, 5]
23	1. Задайте функцию distance(P1, P2), находящую расстояние между точками P1 и P2, каждая из которых задана как кортеж из двух чисел. Используйте библиотечную функцию math:sqrt(). distance({1.0, 2.0}, {0.0, 1.0}) => sqrt(2) (приближённо 1.4142...) 2. Задайте функцию insert(List, X), которая получает отсортированный в порядке возрастания список List и число X, и добавляет X в List так, чтобы снова получить список в порядке возрастания. insert([1, 1.5, 2, 2.5, 3.5], 3) => [1, 1.5, 2, 2.5, 3, 3.5]. 3. Задайте функцию drop_every(List, N), удаляющую каждый N-ый элемент из списка List. drop_every([1,2,3,4,5,6,7], 2) => [1,3,5,7] drop_every([1,2,3,4,5,6,7], 3) => [1,2,4,5,7] 4. Задайте функцию rle_decode(EncodedList), которая работает противоположно функции rle_encode из варианта 1. decode([{a,3},b,{c,2},{a,2}]) => [a,a,a,b,c,c,a,a] 5. Задайте функцию diagonal(Matrix), которая возвращает диагональ матрицы, заданной как список списков. diagonal([[1,2,3], [4,5,6], [7,8,9]]) => [1,5,9]
24	1. Задайте функцию num_roots(A, B, C), находящую число корней квадратного уравнения A*x^2 + B*x + C = 0. num_roots(1, 0, -2) => 2 (т.к. уравнение 1*x^2 + 0*x - 2 = 0 имеет 2 корня) 2. Задайте функцию init(List), возвращающую список List без последнего элемента. init([1,2,3,4]) => [1,2,3] 3. Задайте функцию split(List, N), которая делит список List на две части: первые N элементов и всё, что идёт за ними. split([1, 3, 4, 5], 2) => {[1, 3], [4, 5]} 4. Задайте функцию binary_to_int(Bin), которая переводит двоичную запись числа (в виде строки Bin) в само это число. binary_to_int("100") => 4 binary_to_int("-101") => -5

	5. Задайте функцию sliding_average(List, WindowSize), которая возвращает скользящее среднее списка List с размером окна WindowSize sliding_average([1, 2, 3, 4, 5, 6], 3) => [(1+2+3)/3, (2+3+4)/3, (3+4+5)/3, (4+5+6)/3] == [2.0, 3.0, 4.0, 5.0]
25	1. Задайте функцию ball_volume(R), находящую объём шара радиуса R. Используйте библиотечную функцию math:pi(). ball_volume(1.0) => 4/3*pi (приближённо 4.18879...) 2. Задайте функцию from_to(N, M), строящую список целых чисел от N до M включительно. from_to(1, 4) => [1, 2, 3, 4] 3. Задайте функцию delta(List), которая заменяет каждый элемент списка, кроме первого, на его разность с предыдущим. delta([1,2,4,3]) => [1,1,2,-1] 4. Задайте функцию int_to_binary(N), которая возвращает двоичную запись целого числа N в виде строки (про строки см. lab1.txt) int_to_binary(8) => "1000" int_to_binary(-2) => "-10" 5. Задайте функцию rle_encode(List), которая кодирует список List методом повторов. Если подряд встречается несколько равных элементов, они заменяются на пару {Элемент, Число_повторов}; остальные элементы сохраняются. rle_encode([a,a,a,b,c,c,a,a]) => [{a,3},b,{c,2},{a,2}]

Дополнительные задания:

6. Задайте функцию intersect(List1, List2), находящую все общие элементы двух списков List1 и List2.

intersect([1, 3, 2, 5], [2, 3, 4]) => [3, 2] (или [2, 3]).

intersect([1, 6, 5], [2, 3, 4]) => [].

7. Задайте функцию is_date(DayOfMonth, MonthOfYear, Year), определяющую номер дня недели по числу месяца, номеру месяца и году.

Напомню, что год является високосным, если он либо делится на 4, но не на 100, либо делится на 400.

В качестве точки отсчёта возьмите 1 января 2000 года (суббота). Не используйте каких-то формул для нахождения дня недели, это задание на рекурсию!

is_date(1, 1, 2000) => 6

is_date(1, 2, 2013) => 5

Критерии оценивания

	Задание сдано в срок	**Задание сдано позже**
Задача 1 выполнена верно		
Задача 2 выполнена верно		
Задача 3 выполнена верно		
Задача 4 выполнена верно		
Задача 5 выполнена верно		
Верно выполнено дополнительное задание	1	0,5
Итого	7	3,5