Лабораторная работа 8

1. Загрузить среду программирования.

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

3. Представить результат преподавателю.

Во второй части задачи 2 будьте готовы объяснить, почему сечения расставлены именно так, и как изменится работа программы, если поставить их в других местах.Варианты:

1	1. Определите предикат nu(X, Y), работающий так же, как \+ (X = Y), без использования \+ (используйте комбинацию ! и fail). ?- nu(X, X). No ?- nu(X, foo(Y)). No ?- nu(4, 5). Yes 2. Определите предикат abs(X, Abs), который выполняется, если Abs = |X|, без использования сечений. После этого расставьте зелёные сечения так, чтобы улучшить программу. ?- abs(3, A). A = 3 ?- abs(-1, A). A = 1 3. Определите предикат set_union(List1, List2, Result), собирающий в Result все элементы List1 и все элементы List2. Если элемент лежит одновременно и в List1 и в List2, то в Result он должен входить только один раз. ?- set_union([1,2,3,4,5], [3,5,6,7,8], X). X = [1, 2, 4, 3, 5, 6, 7, 8]
2	1. Определите предикат neq(X, Y), работающий так же, как \+ (X == Y), без использования \+ (используйте комбинацию ! и fail). ?- neq(X, X). No ?- neq(X, Y). Yes ?- neq(4, 5). Yes 2. Определите предикат split(List, Pos, NonPos), который выполняется, если Pos содержит все положительные числа в List в том же порядке, а NonPos содержит все отрицательные числа и 0, без использования сечений. После этого расставьте зелёные сечения так, чтобы улучшить программу. ?- split([1,2,-1,3,0], Pos, NonPos). Pos = [1,2,3], NonPos = [-1, 0] 3. Определите предикат delete_all(List, X, Result), удаляющий все вхождения элемента X из списка List. ?- delete_all([1,2,3,4,3,5], 3, X).

	X = [1, 2, 4, 5]
3	1. Определите предикат my_nonvar(X), работающий так же, как \+ var(X), без использования \+ (используйте комбинацию ! и fail). ?- my_nonvar(X). No ?- my_nonvar(1). Yes ?- my_nonvar(foo(X)). Yes 2. Определите предикат my_ground(Term), выполняющийся, если Term не содержит переменных, без использования сечений. После этого расставьте зелёные сечения так, чтобы улучшить программу. Используйте встроенный предикат =.. (Functor(Arg1, Arg2, ..., ArgN) =.. [Functor, Arg1, Arg2, ..., ArgN]). ?- my_ground(X). No ?- my_ground(foo(X,3)). No ?- my_ground(foo(2,3)). Yes 3. Определите предикат list_add(List, X, Result), который добавляет элемент X к списку List. Если List уже содержит X, то Result должен совпадать с List. ?- list_add([1,2,3,4,3,5], 3, X). X = [1,2,3,4,3,5] ?- list_add([1,2,3,4,3,5], 6, X). X = [1,2,3,4,3,5,6]
4	1. Определите предикат not_member(X, List), истинный, если X не содержится в списке List, без использования \+ (используйте предикат member, и комбинацию ! и fail). ?- not_member([1,2,3,4], 2). No ?- not_member([], 5). Yes 2. Определите предикат delete_first(List, X, Result), удаляющий из List первое вхождение элемента X. ?- delete_first([1,2,3,4,3,5], 3, X). X = [1, 2, 4, 3, 5] 3. Определите предикат set_diff(List1, List2, Result), находящий все элементы списка List1, которые не входят в List2. ?- set_diff([1,2,3,4,3,5], [2,3,6], X). X = [1, 4, 5]
5	1. Определите предикат nu(X, Y), работающий так же, как \+ (X = Y), без использования \+ (используйте комбинацию ! и fail). ?- nu(X, X). No ?- nu(X, foo(Y)). No ?- nu(4, 5). Yes

	2. Определите предикат abs(X, Abs), который выполняется, если Abs = |X|, без использования сечений. После этого расставьте зелёные сечения так, чтобы улучшить программу. ?- abs(3, A). A = 3 ?- abs(-1, A). A = 1 3. Определите предикат set_union(List1, List2, Result), собирающий в Result все элементы List1 и все элементы List2. Если элемент лежит одновременно и в List1 и в List2, то в Result он должен входить только один раз. ?- set_union([1,2,3,4,5], [3,5,6,7,8], X). X = [1, 2, 4, 3, 5, 6, 7, 8]
6	1. Определите предикат neq(X, Y), работающий так же, как \+ (X == Y), без использования \+ (используйте комбинацию ! и fail). ?- neq(X, X). No ?- neq(X, Y). Yes ?- neq(4, 5). Yes 2. Определите предикат split(List, Pos, NonPos), который выполняется, если Pos содержит все положительные числа в List в том же порядке, а NonPos содержит все отрицательные числа и 0, без использования сечений. После этого расставьте зелёные сечения так, чтобы улучшить программу. ?- split([1,2,-1,3,0], Pos, NonPos). Pos = [1,2,3], NonPos = [-1, 0] 3. Определите предикат delete_all(List, X, Result), удаляющий все вхождения элемента X из списка List. ?- delete_all([1,2,3,4,3,5], 3, X). X = [1, 2, 4, 5]
7	1. Определите предикат my_nonvar(X), работающий так же, как \+ var(X), без использования \+ (используйте комбинацию ! и fail). ?- my_nonvar(X). No ?- my_nonvar(1). Yes ?- my_nonvar(foo(X)). Yes 2. Определите предикат my_ground(Term), выполняющийся, если Term не содержит переменных, без использования сечений. После этого расставьте зелёные сечения так, чтобы улучшить программу. Используйте встроенный предикат =.. (Functor(Arg1, Arg2, ..., ArgN) =.. [Functor, Arg1, Arg2, ..., ArgN]). ?- my_ground(X). No ?- my_ground(foo(X,3)). No ?- my_ground(foo(2,3)).

	Yes 3. Определите предикат list_add(List, X, Result), который добавляет элемент X к списку List. Если List уже содержит X, то Result должен совпадать с List. ?- list_add([1,2,3,4,3,5], 3, X). X = [1,2,3,4,3,5] ?- list_add([1,2,3,4,3,5], 6, X). X = [1,2,3,4,3,5,6]
8	1. Определите предикат not_member(X, List), истинный, если X не содержится в списке List, без использования \+ (используйте предикат member, и комбинацию ! и fail). ?- not_member([1,2,3,4], 2). No ?- not_member([], 5). Yes 2. Определите предикат delete_first(List, X, Result), удаляющий из List первое вхождение элемента X. ?- delete_first([1,2,3,4,3,5], 3, X). X = [1, 2, 4, 3, 5] 3. Определите предикат set_diff(List1, List2, Result), находящий все элементы списка List1, которые не входят в List2. ?- set_diff([1,2,3,4,3,5], [2,3,6], X). X = [1, 4, 5]
9	1. Определите предикат nu(X, Y), работающий так же, как \+ (X = Y), без использования \+ (используйте комбинацию ! и fail). ?- nu(X, X). No ?- nu(X, foo(Y)). No ?- nu(4, 5). Yes 2. Определите предикат abs(X, Abs), который выполняется, если Abs = |X|, без использования сечений. После этого расставьте зелёные сечения так, чтобы улучшить программу. ?- abs(3, A). A = 3 ?- abs(-1, A). A = 1 3. Определите предикат set_union(List1, List2, Result), собирающий в Result все элементы List1 и все элементы List2. Если элемент лежит одновременно и в List1 и в List2, то в Result он должен входить только один раз. ?- set_union([1,2,3,4,5], [3,5,6,7,8], X). X = [1, 2, 4, 3, 5, 6, 7, 8]
10	1. Определите предикат neq(X, Y), работающий так же, как \+ (X == Y), без использования \+ (используйте комбинацию ! и fail). ?- neq(X, X). No ?- neq(X, Y).

	Yes ?- neq(4, 5). Yes 2. Определите предикат split(List, Pos, NonPos), который выполняется, если Pos содержит все положительные числа в List в том же порядке, а NonPos содержит все отрицательные числа и 0, без использования сечений. После этого расставьте зелёные сечения так, чтобы улучшить программу. ?- split([1,2,-1,3,0], Pos, NonPos). Pos = [1,2,3], NonPos = [-1, 0] 3. Определите предикат delete_all(List, X, Result), удаляющий все вхождения элемента X из списка List. ?- delete_all([1,2,3,4,3,5], 3, X). X = [1, 2, 4, 5]
11	1. Определите предикат my_nonvar(X), работающий так же, как \+ var(X), без использования \+ (используйте комбинацию ! и fail). ?- my_nonvar(X). No ?- my_nonvar(1). Yes ?- my_nonvar(foo(X)). Yes 2. Определите предикат my_ground(Term), выполняющийся, если Term не содержит переменных, без использования сечений. После этого расставьте зелёные сечения так, чтобы улучшить программу. Используйте встроенный предикат =.. (Functor(Arg1, Arg2, ..., ArgN) =.. [Functor, Arg1, Arg2, ..., ArgN]). ?- my_ground(X). No ?- my_ground(foo(X,3)). No ?- my_ground(foo(2,3)). Yes 3. Определите предикат list_add(List, X, Result), который добавляет элемент X к списку List. Если List уже содержит X, то Result должен совпадать с List. ?- list_add([1,2,3,4,3,5], 3, X). X = [1,2,3,4,3,5] ?- list_add([1,2,3,4,3,5], 6, X). X = [1,2,3,4,3,5,6]
12	1. Определите предикат not_member(X, List), истинный, если X не содержится в списке List, без использования \+ (используйте предикат member, и комбинацию ! и fail). ?- not_member([1,2,3,4], 2). No ?- not_member([], 5). Yes 2. Определите предикат delete_first(List, X, Result), удаляющий из List первое вхождение элемента X. ?- delete_first([1,2,3,4,3,5], 3, X).

	X = [1, 2, 4, 3, 5] 3. Определите предикат set_diff(List1, List2, Result), находящий все элементы списка List1, которые не входят в List2. ?- set_diff([1,2,3,4,3,5], [2,3,6], X). X = [1, 4, 5]
13	1. Определите предикат nu(X, Y), работающий так же, как \+ (X = Y), без использования \+ (используйте комбинацию ! и fail). ?- nu(X, X). No ?- nu(X, foo(Y)). No ?- nu(4, 5). Yes 2. Определите предикат abs(X, Abs), который выполняется, если Abs = |X|, без использования сечений. После этого расставьте зелёные сечения так, чтобы улучшить программу. ?- abs(3, A). A = 3 ?- abs(-1, A). A = 1 3. Определите предикат set_union(List1, List2, Result), собирающий в Result все элементы List1 и все элементы List2. Если элемент лежит одновременно и в List1 и в List2, то в Result он должен входить только один раз. ?- set_union([1,2,3,4,5], [3,5,6,7,8], X). X = [1, 2, 4, 3, 5, 6, 7, 8]
14	1. Определите предикат neq(X, Y), работающий так же, как \+ (X == Y), без использования \+ (используйте комбинацию ! и fail). ?- neq(X, X). No ?- neq(X, Y). Yes ?- neq(4, 5). Yes 2. Определите предикат split(List, Pos, NonPos), который выполняется, если Pos содержит все положительные числа в List в том же порядке, а NonPos содержит все отрицательные числа и 0, без использования сечений. После этого расставьте зелёные сечения так, чтобы улучшить программу. ?- split([1,2,-1,3,0], Pos, NonPos). Pos = [1,2,3], NonPos = [-1, 0] 3. Определите предикат delete_all(List, X, Result), удаляющий все вхождения элемента X из списка List. ?- delete_all([1,2,3,4,3,5], 3, X). X = [1, 2, 4, 5]
15	1. Определите предикат my_nonvar(X), работающий так же, как \+ var(X), без использования \+ (используйте комбинацию ! и fail). ?- my_nonvar(X). No

	?- my_nonvar(1). Yes ?- my_nonvar(foo(X)). Yes 2. Определите предикат my_ground(Term), выполняющийся, если Term не содержит переменных, без использования сечений. После этого расставьте зелёные сечения так, чтобы улучшить программу. Используйте встроенный предикат =.. (Functor(Arg1, Arg2, ..., ArgN) =.. [Functor, Arg1, Arg2, ..., ArgN]). ?- my_ground(X). No ?- my_ground(foo(X,3)). No ?- my_ground(foo(2,3)). Yes 3. Определите предикат list_add(List, X, Result), который добавляет элемент X к списку List. Если List уже содержит X, то Result должен совпадать с List. ?- list_add([1,2,3,4,3,5], 3, X). X = [1,2,3,4,3,5] ?- list_add([1,2,3,4,3,5], 6, X). X = [1,2,3,4,3,5,6]
16	1. Определите предикат not_member(X, List), истинный, если X не содержится в списке List, без использования \+ (используйте предикат member, и комбинацию ! и fail). ?- not_member([1,2,3,4], 2). No ?- not_member([], 5). Yes 2. Определите предикат delete_first(List, X, Result), удаляющий из List первое вхождение элемента X. ?- delete_first([1,2,3,4,3,5], 3, X). X = [1, 2, 4, 3, 5] 3. Определите предикат set_diff(List1, List2, Result), находящий все элементы списка List1, которые не входят в List2. ?- set_diff([1,2,3,4,3,5], [2,3,6], X). X = [1, 4, 5]
17	1. Определите предикат nu(X, Y), работающий так же, как \+ (X = Y), без использования \+ (используйте комбинацию ! и fail). ?- nu(X, X). No ?- nu(X, foo(Y)). No ?- nu(4, 5). Yes 2. Определите предикат abs(X, Abs), который выполняется, если Abs = |X|, без использования сечений. После этого расставьте зелёные сечения так, чтобы улучшить программу. ?- abs(3, A).

	A = 3 ?- abs(-1, A). A = 1 3. Определите предикат set_union(List1, List2, Result), собирающий в Result все элементы List1 и все элементы List2. Если элемент лежит одновременно и в List1 и в List2, то в Result он должен входить только один раз. ?- set_union([1,2,3,4,5], [3,5,6,7,8], X). X = [1, 2, 4, 3, 5, 6, 7, 8]
18	1. Определите предикат neq(X, Y), работающий так же, как \+ (X == Y), без использования \+ (используйте комбинацию ! и fail). ?- neq(X, X). No ?- neq(X, Y). Yes ?- neq(4, 5). Yes 2. Определите предикат split(List, Pos, NonPos), который выполняется, если Pos содержит все положительные числа в List в том же порядке, а NonPos содержит все отрицательные числа и 0, без использования сечений. После этого расставьте зелёные сечения так, чтобы улучшить программу. ?- split([1,2,-1,3,0], Pos, NonPos). Pos = [1,2,3], NonPos = [-1, 0] 3. Определите предикат delete_all(List, X, Result), удаляющий все вхождения элемента X из списка List. ?- delete_all([1,2,3,4,3,5], 3, X). X = [1, 2, 4, 5]
19	1. Определите предикат my_nonvar(X), работающий так же, как \+ var(X), без использования \+ (используйте комбинацию ! и fail). ?- my_nonvar(X). No ?- my_nonvar(1). Yes ?- my_nonvar(foo(X)). Yes 2. Определите предикат my_ground(Term), выполняющийся, если Term не содержит переменных, без использования сечений. После этого расставьте зелёные сечения так, чтобы улучшить программу. Используйте встроенный предикат =.. (Functor(Arg1, Arg2, ..., ArgN) =.. [Functor, Arg1, Arg2, ..., ArgN]). ?- my_ground(X). No ?- my_ground(foo(X,3)). No ?- my_ground(foo(2,3)). Yes 3. Определите предикат list_add(List, X, Result), который добавляет элемент X к списку List. Если List уже содержит X, то Result должен совпадать с List. ?- list_add([1,2,3,4,3,5], 3, X).

	X = [1,2,3,4,3,5] ?- list_add([1,2,3,4,3,5], 6, X). X = [1,2,3,4,3,5,6]
20	1. Определите предикат not_member(X, List), истинный, если X не содержится в списке List, без использования \+ (используйте предикат member, и комбинацию ! и fail). ?- not_member([1,2,3,4], 2). No ?- not_member([], 5). Yes 2. Определите предикат delete_first(List, X, Result), удаляющий из List первое вхождение элемента X. ?- delete_first([1,2,3,4,3,5], 3, X). X = [1, 2, 4, 3, 5] 3. Определите предикат set_diff(List1, List2, Result), находящий все элементы списка List1, которые не входят в List2. ?- set_diff([1,2,3,4,3,5], [2,3,6], X). X = [1, 4, 5]
21	1. Определите предикат nu(X, Y), работающий так же, как \+ (X = Y), без использования \+ (используйте комбинацию ! и fail). ?- nu(X, X). No ?- nu(X, foo(Y)). No ?- nu(4, 5). Yes 2. Определите предикат abs(X, Abs), который выполняется, если Abs = |X|, без использования сечений. После этого расставьте зелёные сечения так, чтобы улучшить программу. ?- abs(3, A). A = 3 ?- abs(-1, A). A = 1 3. Определите предикат set_union(List1, List2, Result), собирающий в Result все элементы List1 и все элементы List2. Если элемент лежит одновременно и в List1 и в List2, то в Result он должен входить только один раз. ?- set_union([1,2,3,4,5], [3,5,6,7,8], X). X = [1, 2, 4, 3, 5, 6, 7, 8]
22	1. Определите предикат neq(X, Y), работающий так же, как \+ (X == Y), без использования \+ (используйте комбинацию ! и fail). ?- neq(X, X). No ?- neq(X, Y). Yes ?- neq(4, 5). Yes

	2. Определите предикат split(List, Pos, NonPos), который выполняется, если Pos содержит все положительные числа в List в том же порядке, а NonPos содержит все отрицательные числа и 0, без использования сечений. После этого расставьте зелёные сечения так, чтобы улучшить программу. ?- split([1,2,-1,3,0], Pos, NonPos). Pos = [1,2,3], NonPos = [-1, 0] 3. Определите предикат delete_all(List, X, Result), удаляющий все вхождения элемента X из списка List. ?- delete_all([1,2,3,4,3,5], 3, X). X = [1, 2, 4, 5]
23	1. Определите предикат my_nonvar(X), работающий так же, как \+ var(X), без использования \+ (используйте комбинацию ! и fail). ?- my_nonvar(X). No ?- my_nonvar(1). Yes ?- my_nonvar(foo(X)). Yes 2. Определите предикат my_ground(Term), выполняющийся, если Term не содержит переменных, без использования сечений. После этого расставьте зелёные сечения так, чтобы улучшить программу. Используйте встроенный предикат =.. (Functor(Arg1, Arg2, ..., ArgN) =.. [Functor, Arg1, Arg2, ..., ArgN]). ?- my_ground(X). No ?- my_ground(foo(X,3)). No ?- my_ground(foo(2,3)). Yes 3. Определите предикат list_add(List, X, Result), который добавляет элемент X к списку List. Если List уже содержит X, то Result должен совпадать с List. ?- list_add([1,2,3,4,3,5], 3, X). X = [1,2,3,4,3,5] ?- list_add([1,2,3,4,3,5], 6, X). X = [1,2,3,4,3,5,6]
24	1. Определите предикат not_member(X, List), истинный, если X не содержится в списке List, без использования \+ (используйте предикат member, и комбинацию ! и fail). ?- not_member([1,2,3,4], 2). No ?- not_member([], 5). Yes 2. Определите предикат delete_first(List, X, Result), удаляющий из List первое вхождение элемента X. ?- delete_first([1,2,3,4,3,5], 3, X). X = [1, 2, 4, 3, 5] 3. Определите предикат set_diff(List1, List2, Result), находящий все элементы списка List1, которые не входят в List2.

	?- set_diff([1,2,3,4,3,5], [2,3,6], X). X = [1, 4, 5]
25	1. Определите предикат nu(X, Y), работающий так же, как \+ (X = Y), без использования \+ (используйте комбинацию ! и fail). ?- nu(X, X). No ?- nu(X, foo(Y)). No ?- nu(4, 5). Yes 2. Определите предикат abs(X, Abs), который выполняется, если Abs = |X|, без использования сечений. После этого расставьте зелёные сечения так, чтобы улучшить программу. ?- abs(3, A). A = 3 ?- abs(-1, A). A = 1 3. Определите предикат set_union(List1, List2, Result), собирающий в Result все элементы List1 и все элементы List2. Если элемент лежит одновременно и в List1 и в List2, то в Result он должен входить только один раз. ?- set_union([1,2,3,4,5], [3,5,6,7,8], X). X = [1, 2, 4, 3, 5, 6, 7, 8]

Дополнительные задания:

4. Вернитесь к решениям лабораторных 6 и 7. Найдите, где можно расставить зелёные сечения для улучшения эффективности. Кроме того, возможно, что в некоторых случаях Prolog выдаёт неправильные дополнительные решения после отката (особенно в задачах 3 и 4 лабораторной 6). Если найдёте такие случаи, исправьте их с использованием сечений.

5. Определите предикат unifiable(List, Term, Result), который возвращает список всех элементов List, которые можно сопоставить с Term, но _без_ конкретизации входящих переменных.

?- unifiable([X, b, t(Y)], t(a), List].

List = [X, t(Y)] %% X = t(a) и t(Y) = t(a) удаются, b = t(a) -- нет

Подсказка: подумайте, чем отличаются \+(\+(X = Y)) и X = Y.

Критерии оценивания

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