Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОННОЙ ТЕХНИКИ»

Институт системной и программной инженерии

и информационных технологий (СПИНТех)

**Отчёт**

по дисциплине «Функциональное и логическое программирование»

**Лабораторная работа №6**

**Вариант-6**

Руководитель

\_\_\_\_\_\_\_\_\_\_ Петрова A. В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г.

Студент группы ПИН-33

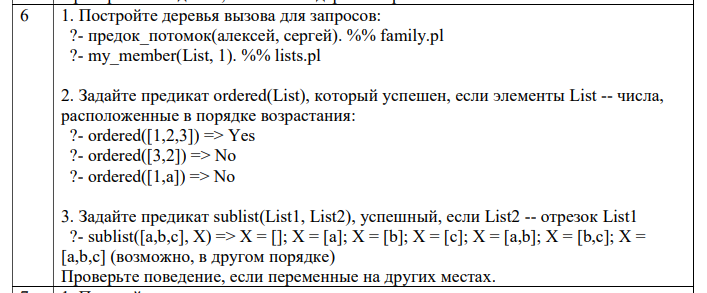
\_\_\_\_\_\_\_\_\_\_ Исламов Р. Р.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г.

*Москва*

*2024*

# Задание



# 1.1 Листинг

family.pl

предок\_потомок(алексей, сергей).

предок\_потомок(сергей, петр).

предок\_потомок(сергей, мария).

предок\_потомок(петр, анна).

предок\_потомок(мария, елена).

предок\_потомок(петр, василий).

предок\_потомок(алексей, екатерина).

% Тесты

:- begin\_tests(family).

test(предок\_потомок\_существует) :-

предок\_потомок(алексей, сергей),

предок\_потомок(сергей, петр).

test(предок\_потомок\_не\_существует, fail) :-

предок\_потомок(алексей, анна).

:- end\_tests(family).

lists.pl

my\_member(Element, [Element|\_]).

my\_member(Element, [\_|Tail]) :-

my\_member(Element, Tail).

my\_length([], 0).

my\_length([\_|Tail], Length) :-

my\_length(Tail, TailLength),

Length is TailLength + 1.

my\_append([], List, List).

my\_append([Head|Tail1], List2, [Head|Result]) :-

my\_append(Tail1, List2, Result).

% Тесты

:- begin\_tests(lists).

test(член\_списка\_существует) :-

my\_member(1, [3, 2, 1, 4]),

!.

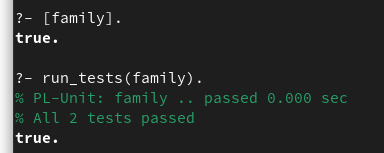
test(член\_списка\_не\_существует, fail) :-

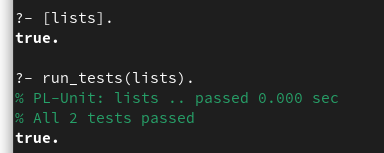
my\_member(5, [3, 2, 1, 4]),

!.

:- end\_tests(lists).

# 1.2 Результат работы





# 2.1 Листинг

ordered([]).

ordered([\_]).

ordered([X,Y|T]) :-

number(X),

number(Y),

X =< Y,

ordered([Y|T]). % Рекурсивно проверяем элементы

% Тесты

:- begin\_tests(ordered).

test(ordered\_empty) :-

ordered([]),

!.

test(ordered\_single\_element) :-

ordered([1]),

!.

test(ordered\_sorted\_list) :-

ordered([1, 2, 3]),

!.

test(ordered\_unsorted\_list) :-

\+ ordered([3, 2]),

!.

test(ordered\_mixed\_list) :-

\+ ordered([1, a]),

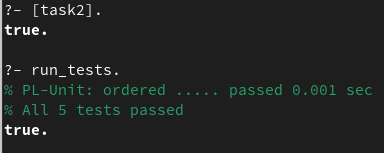
!.

:- end\_tests(ordered).

run\_tests :-

run\_tests(ordered).

# 2.2 Результат работы



# 3.1 Листинг

remove\_duplicates([], []).

remove\_duplicates([X|Xs], Ys) :-

member(X, Xs),

!,

remove\_duplicates(Xs, Ys).

remove\_duplicates([X|Xs], [X|Ys]) :-

remove\_duplicates(Xs, Ys).

:- begin\_tests(remove\_duplicates).

test(empty\_list) :-

remove\_duplicates([], []),

!.

test(no\_duplicates) :-

remove\_duplicates([a, b, c], [a, b, c]),

!.

test(duplicates\_removed) :-

remove\_duplicates([a, b, a, b, c, c, a], X),

permutation(X, [a, b, c]),

!.

test(variables\_in\_different\_places) :-

remove\_duplicates([a, b, c], X),

X = [a, b, c],

!.

test(duplicates\_with\_variables\_in\_different\_places) :-

remove\_duplicates([a, b, a, b, c, c, a], X),

permutation(X, [a, b, c]),

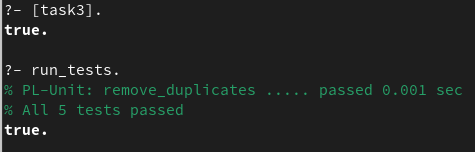
!.

:- end\_tests(remove\_duplicates).

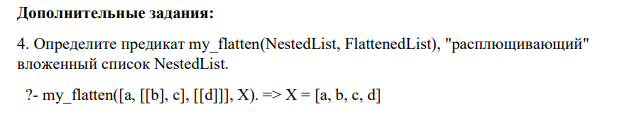
run\_tests :-

run\_tests(remove\_duplicates).

# 3.2 Результат работы



# 4.1 Листинг



my\_flatten([], []). % Базовый случай

my\_flatten([Head|Tail], Flattened) :-

is\_list(Head),

!,

my\_flatten(Head, FlattenedHead),

my\_flatten(Tail, FlattenedTail),

append(FlattenedHead, FlattenedTail, Flattened).

my\_flatten([Head|Tail], [Head|FlattenedTail]) :-

% Рекурсивно разбиваем оставшуюся часть

\+ is\_list(Head),

my\_flatten(Tail, FlattenedTail).

:- begin\_tests(my\_flatten).

% Тесты

test(empty\_list) :-

my\_flatten([], []),

!.

test(single\_flat\_list) :-

my\_flatten([a, b, c], [a, b, c]),

!.

test(nested\_list) :-

my\_flatten([a, [[b], c], [[d]]], [a, b, c, d]),

!.

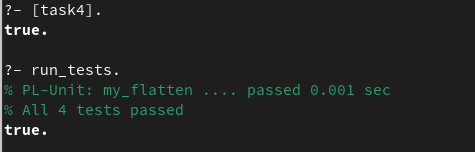
test(mixed\_list) :-

my\_flatten([a, [b, [c, [d]], e]], [a, b, c, d, e]),

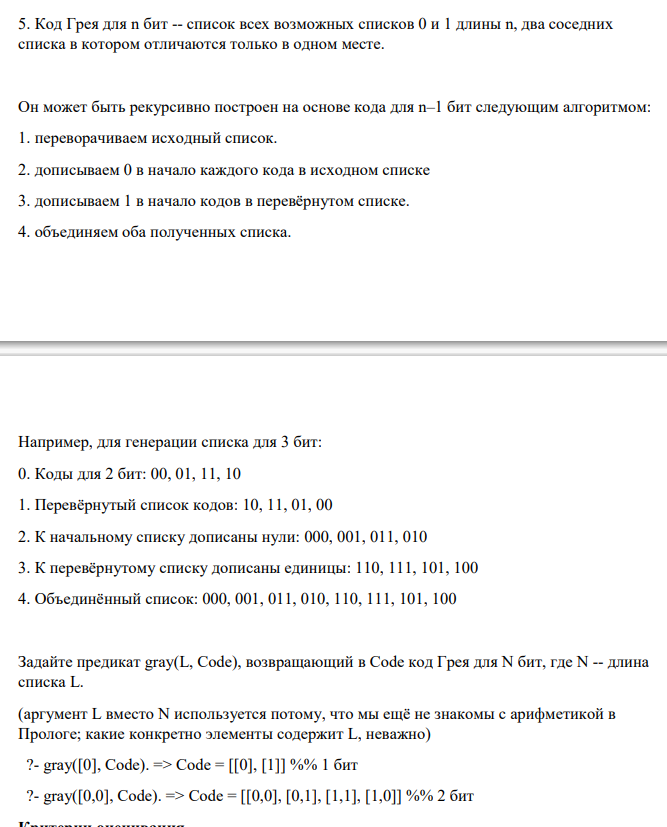
!.

:- end\_tests(my\_flatten).

# 4.2 Результат работы



# 5.1 Листинг



gray([0], [[0], [1]]).

gray(L, Code) :-

length(L, Len),

Len > 1,

reverse(L, RevL),

append\_zero(L, LWithZero),

append\_one(RevL, LWithOne),

append(LWithZero, LWithOne, Code).

append\_zero([], []).

append\_zero([H|T], [[0|H]|T1]) :-

append\_zero(T, T1).

append\_one([], []).

append\_one([H|T], [[1|H]|T1]) :-

append\_one(T, T1).

:- begin\_tests(gray).

test(one\_bit) :-

gray([0], Code),

reverse(Code, Expected),

permutation(Code, Expected),

!.

test(two\_bits) :-

gray([0, 0], Code),

reverse(Code, Expected),

permutation(Code, Expected),

!.

test(three\_bits) :-

gray([0, 0, 0], Code),

reverse(Code, Expected),

permutation(Code, Expected),

!.

test(mixed\_bits) :-

gray([1, 0, 1, 0], Code),

reverse(Code, Expected),

permutation(Code, Expected),

!.

:- end\_tests(gray).

# 5.2 Результат работы

