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 №3**

**Вариант-6**

Руководитель

\_\_\_\_\_\_\_\_\_\_ Петрова A. В.

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

Студент группы ПИН-33

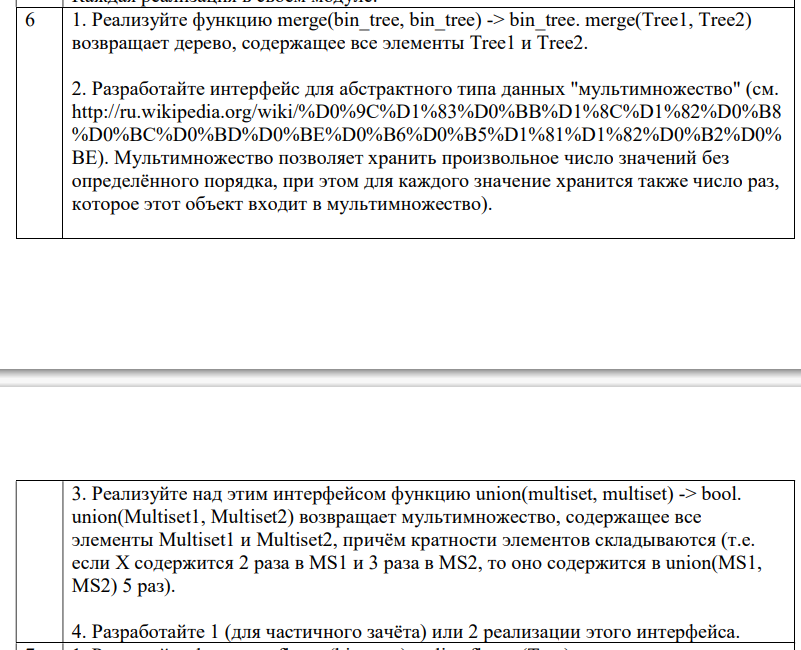
\_\_\_\_\_\_\_\_\_\_ Исламов Р. Р.

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

*Москва*

*2024*

# Задание



# 1.1 Листинг

-module(task1).

-export([merge/2, insert/2, in\_order\_traversal/1]).

-record(node, {value, left = nil, right = nil}).

%% Объединение двух деревьев

merge(Tree1, nil) ->

    Tree1;

merge(nil, Tree2) ->

    Tree2;

merge(Tree1, Tree2) ->

    %% Получаем значения и вставляем каждое значение

    Values1 = in\_order\_traversal(Tree1),

    MergedTree = lists:foldl(fun(Value, AccTree) ->

                                insert(Value, AccTree)

                             end, Tree2, Values1),

    MergedTree.

%% Вставка элемента в бинарное дерево

insert(Value, nil) ->

    #node{value = Value};

insert(Value, #node{value = NodeValue, left = Left, right = Right} = Node) ->

    case Value < NodeValue of

        true ->

            Node#node{left = insert(Value, Left)};

        false ->

            Node#node{right = insert(Value, Right)}

    end.

%% Обход дерева

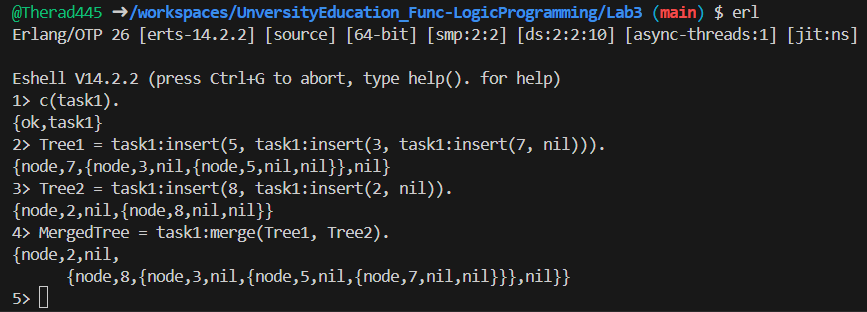
in\_order\_traversal(nil) ->

    [];

in\_order\_traversal(#node{value = Value, left = Left, right = Right}) ->

    in\_order\_traversal(Left) ++ [Value] ++ in\_order\_traversal(Right).

# 1.2 Результат работы



# 2.1 Листинг

-module(task2).

-export([new/0, add/2, remove/2, count/2, elements/1]).

-record(multiset, {map = #{}}).

new() ->

    #multiset{map = #{}}.

add(Element, Multiset) ->

    UpdatedMap = maps:update\_with(Element, fun(Count) -> Count + 1 end, 1, Multiset#multiset.map),

    Multiset#multiset{map = UpdatedMap}.

remove(Element, Multiset) ->

    case maps:is\_key(Element, Multiset#multiset.map) of

        true ->

            Count = maps:get(Element, Multiset#multiset.map),

            case Count of

                1 ->

                    UpdatedMap = maps:remove(Element, Multiset#multiset.map),

                    Multiset#multiset{map = UpdatedMap};

                \_ ->

                    UpdatedMap = maps:update\_with(Element, fun(Count) -> Count - 1 end, Multiset#multiset.map),

                    Multiset#multiset{map = UpdatedMap}

            end;

        false ->

            Multiset

    end.

count(Element, Multiset) ->

    maps:get(Element, Multiset#multiset.map, 0).

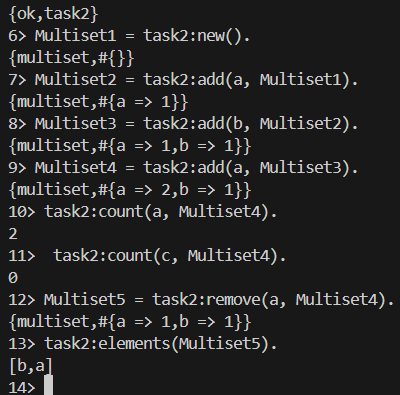
elements(Multiset) ->

    maps:fold(fun(Element, Count, Acc) ->

                   lists:duplicate(Count, Element) ++ Acc

               end, [], Multiset#multiset.map).

# 2.2 Результат работы



# 3.1 Листинг

union(Multiset1, Multiset2) ->

    Elements1 = maps:to\_list(Multiset1#multiset.map),

    Elements2 = maps:to\_list(Multiset2#multiset.map),

    AllElements = Elements1 ++ Elements2,

    UnionMultiset = lists:foldl(fun({Element, Count}, AccMultiset) ->

                                      UpdatedMultiset = add(Element, AccMultiset),

                                      lists:foldl(fun(\_Index, Acc) ->

                                                        UpdatedMultiset1 = add(Element, UpdatedMultiset),

                                                        Acc + 1

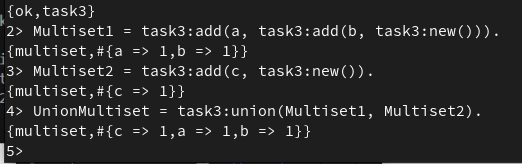
                                                end, 0, lists:seq(1, Count)),

                                      UpdatedMultiset

                              end, new(), AllElements),

    UnionMultiset.

# 3.2 Результат работы



# 4.1 Листинг

union(Multiset1, Multiset2) ->

AllPairs = Multiset1#multiset.pairs ++ Multiset2#multiset.pairs,

SortedPairs = lists:usort(AllPairs),

UnionMultiset = lists:foldl(fun({Element, Count}, AccMultiset) ->

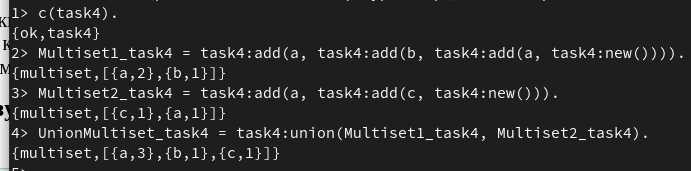
UpdatedMultiset = add(Element, AccMultiset, Count),

UpdatedMultiset

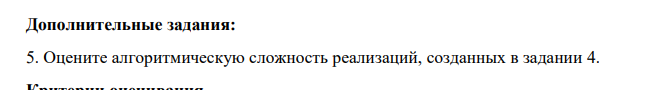
end, new(), SortedPairs),

UnionMultiset.

# 4.2 Результат работы



# 5 Сложность алгоритмов



Итоговая сложность:

Добавление (`add/2`), Удаление (`remove/2`), Подсчет количества вхождений элемента (`count/2`): O(1) в среднем случае.

Получение всех элементов (`elements/1`): O(n)

Реализация с «maps»:

Объединение мультимножеств (`union/2`): O((n1 + n2) \* m), где n1 и n2 - количество элементов в объединяемых мультимножество, а m - максимальная кратность элемента в объединенном мультимножестве.

Реализация с использованием списка:

Объединение мультимножеств (`union/2`): O((n1 + n2)^2), где n1 и n2 - количество элементов в объединяемых мультимножествах.

В среднем случае реализация с `maps` имеет эффективнее.