Лабораторная работа 3

1. Загрузить среду программирования.

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

3. Представить результат преподавателю.

Варианты:

1	1. Реализуйте функцию split(bin_tree, any) -> {bin_tree, bin_tree}. split(Tree, X) возвращает пару {TreeLT, TreeGT}; TreeLT содержит все элементы Tree, меньшие X, а TreeGT -- все элементы, большие X. 2. Разработайте интерфейс для абстрактного типа данных "множество" (см. http://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE_%28%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%29). Множество позволяет хранить произвольное число значений без определённого порядка, при этом каждое значение хранится не более одного раза. 3. Реализуйте над этим интерфейсом функцию is_subset(set, set) -> bool. is_subset(Set1, Set2) возвращает true, если Set1 -- подмножество Set2 (т.е. все элементы Set1 содержатся в Set2). 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса. Каждая реализация в своём модуле.
2	1. Реализуйте функцию merge(bin_tree, bin_tree) -> bin_tree. merge(Tree1, Tree2) возвращает дерево, содержащее все элементы Tree1 и Tree2. 2. Разработайте интерфейс для абстрактного типа данных "мультимножество" (см. http://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE). Мультимножество позволяет хранить произвольное число значений без определённого порядка, при этом для каждого значение хранится также число раз, которое этот объект входит в мультимножество). 3. Реализуйте над этим интерфейсом функцию union(multiset, multiset) -> bool. union(Multiset1, Multiset2) возвращает мультимножество, содержащее все элементы Multiset1 и Multiset2, причём кратности элементов складываются (т.е. если X содержится 2 раза в MS1 и 3 раза в MS2, то оно содержится в union(MS1, MS2) 5 раз). 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса.
3	1. Реализуйте функцию flatten(bin_tree) -> list. flatten(Tree) возвращает список всех данных в дереве в порядке возрастания. 2. Разработайте интерфейс для абстрактного типа данных "словарь" (см. http://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2). Словарь позволяет хранить

	произвольное число пар ключ-значение без определённого порядка, при этом две пары с одним ключом одновременно не допускаются. 3. Реализуйте над этим интерфейсом функцию all_values(dictionary) -> [any]. all_values(Dict) возвращает список всех значений в словаре. 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса.
4	1. Реализуйте функцию count_leaves(bin_tree) -> integer. count_leaves(Tree) возвращает число листьев дерева (т.е. вершин, у которых оба поддерева -- пустые). 2. Разработайте интерфейс для абстрактного типа данных "очередь с приоритетом" (см. http://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C_%D1%81_%D0%BF%D1%80%D0%B8%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82%D0%BE%D0%BC). Очередь с приоритетом позволяет хранить пары (значение, приоритет), при этом каждое значение может храниться несколько раз (в том числе с одинаковым приоритетом) и поддерживает операцию извлечения пары с минимальным приоритетом. 3. Реализуйте над этим интерфейсом функцию to_list(priority_queue) -> [any]. to_list(Queue) возвращает список всех значений, содержащихся в Queue, в порядке возрастания приоритета. 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса.
5	1. Реализуйте функцию split(bin_tree, any) -> {bin_tree, bin_tree}. split(Tree, X) возвращает пару {TreeLT, TreeGT}; TreeLT содержит все элементы Tree, меньшие X, а TreeGT -- все элементы, большие X. 2. Разработайте интерфейс для абстрактного типа данных "множество" (см. http://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE_%28%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%29). Множество позволяет хранить произвольное число значений без определённого порядка, при этом каждое значение хранится не более одного раза. 3. Реализуйте над этим интерфейсом функцию is_subset(set, set) -> bool. is_subset(Set1, Set2) возвращает true, если Set1 -- подмножество Set2 (т.е. все элементы Set1 содержатся в Set2). 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса. Каждая реализация в своём модуле.
6	1. Реализуйте функцию merge(bin_tree, bin_tree) -> bin_tree. merge(Tree1, Tree2) возвращает дерево, содержащее все элементы Tree1 и Tree2. 2. Разработайте интерфейс для абстрактного типа данных "мультимножество" (см. http://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE). Мультимножество позволяет хранить произвольное число значений без определённого порядка, при этом для каждого значение хранится также число раз, которое этот объект входит в мультимножество).

	3. Реализуйте над этим интерфейсом функцию union(multiset, multiset) -> bool. union(Multiset1, Multiset2) возвращает мультимножество, содержащее все элементы Multiset1 и Multiset2, причём кратности элементов складываются (т.е. если X содержится 2 раза в MS1 и 3 раза в MS2, то оно содержится в union(MS1, MS2) 5 раз). 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса.
7	1. Реализуйте функцию flatten(bin_tree) -> list. flatten(Tree) возвращает список всех данных в дереве в порядке возрастания. 2. Разработайте интерфейс для абстрактного типа данных "словарь" (см. http://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2). Словарь позволяет хранить произвольное число пар ключ-значение без определённого порядка, при этом две пары с одним ключом одновременно не допускаются. 3. Реализуйте над этим интерфейсом функцию all_values(dictionary) -> [any]. all_values(Dict) возвращает список всех значений в словаре. 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса.
8	1. Реализуйте функцию count_leaves(bin_tree) -> integer. count_leaves(Tree) возвращает число листьев дерева (т.е. вершин, у которых оба поддерева -- пустые). 2. Разработайте интерфейс для абстрактного типа данных "очередь с приоритетом" (см. http://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C_%D1%81_%D0%BF%D1%80%D0%B8%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82%D0%BE%D0%BC). Очередь с приоритетом позволяет хранить пары (значение, приоритет), при этом каждое значение может храниться несколько раз (в том числе с одинаковым приоритетом) и поддерживает операцию извлечения пары с минимальным приоритетом. 3. Реализуйте над этим интерфейсом функцию to_list(priority_queue) -> [any]. to_list(Queue) возвращает список всех значений, содержащихся в Queue, в порядке возрастания приоритета. 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса.
9	1. Реализуйте функцию split(bin_tree, any) -> {bin_tree, bin_tree}. split(Tree, X) возвращает пару {TreeLT, TreeGT}; TreeLT содержит все элементы Tree, меньшие X, а TreeGT -- все элементы, большие X. 2. Разработайте интерфейс для абстрактного типа данных "множество" (см. http://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE_%28%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%29). Множество позволяет хранить произвольное число значений без определённого порядка, при этом каждое значение хранится не более одного раза. 3. Реализуйте над этим интерфейсом функцию is_subset(set, set) -> bool. is_subset(Set1, Set2) возвращает true, если Set1 -- подмножество Set2 (т.е. все элементы Set1 содержатся в Set2).

	4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса. Каждая реализация в своём модуле.
10	1. Реализуйте функцию merge(bin_tree, bin_tree) -> bin_tree. merge(Tree1, Tree2) возвращает дерево, содержащее все элементы Tree1 и Tree2. 2. Разработайте интерфейс для абстрактного типа данных "мультимножество" (см. http://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE). Мультимножество позволяет хранить произвольное число значений без определённого порядка, при этом для каждого значение хранится также число раз, которое этот объект входит в мультимножество). 3. Реализуйте над этим интерфейсом функцию union(multiset, multiset) -> bool. union(Multiset1, Multiset2) возвращает мультимножество, содержащее все элементы Multiset1 и Multiset2, причём кратности элементов складываются (т.е. если X содержится 2 раза в MS1 и 3 раза в MS2, то оно содержится в union(MS1, MS2) 5 раз). 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса.
11	1. Реализуйте функцию flatten(bin_tree) -> list. flatten(Tree) возвращает список всех данных в дереве в порядке возрастания. 2. Разработайте интерфейс для абстрактного типа данных "словарь" (см. http://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2). Словарь позволяет хранить произвольное число пар ключ-значение без определённого порядка, при этом две пары с одним ключом одновременно не допускаются. 3. Реализуйте над этим интерфейсом функцию all_values(dictionary) -> [any]. all_values(Dict) возвращает список всех значений в словаре. 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса.
12	1. Реализуйте функцию count_leaves(bin_tree) -> integer. count_leaves(Tree) возвращает число листьев дерева (т.е. вершин, у которых оба поддерева -- пустые). 2. Разработайте интерфейс для абстрактного типа данных "очередь с приоритетом" (см. http://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C_%D1%81_%D0%BF%D1%80%D0%B8%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82%D0%BE%D0%BC). Очередь с приоритетом позволяет хранить пары (значение, приоритет), при этом каждое значение может храниться несколько раз (в том числе с одинаковым приоритетом) и поддерживает операцию извлечения пары с минимальным приоритетом. 3. Реализуйте над этим интерфейсом функцию to_list(priority_queue) -> [any]. to_list(Queue) возвращает список всех значений, содержащихся в Queue, в порядке возрастания приоритета. 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса.

13	1. Реализуйте функцию split(bin_tree, any) -> {bin_tree, bin_tree}. split(Tree, X) возвращает пару {TreeLT, TreeGT}; TreeLT содержит все элементы Tree, меньшие X, а TreeGT -- все элементы, большие X. 2. Разработайте интерфейс для абстрактного типа данных "множество" (см. http://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE_%28%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%29). Множество позволяет хранить произвольное число значений без определённого порядка, при этом каждое значение хранится не более одного раза. 3. Реализуйте над этим интерфейсом функцию is_subset(set, set) -> bool. is_subset(Set1, Set2) возвращает true, если Set1 -- подмножество Set2 (т.е. все элементы Set1 содержатся в Set2). 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса. Каждая реализация в своём модуле.
14	1. Реализуйте функцию merge(bin_tree, bin_tree) -> bin_tree. merge(Tree1, Tree2) возвращает дерево, содержащее все элементы Tree1 и Tree2. 2. Разработайте интерфейс для абстрактного типа данных "мультимножество" (см. http://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE). Мультимножество позволяет хранить произвольное число значений без определённого порядка, при этом для каждого значение хранится также число раз, которое этот объект входит в мультимножество). 3. Реализуйте над этим интерфейсом функцию union(multiset, multiset) -> bool. union(Multiset1, Multiset2) возвращает мультимножество, содержащее все элементы Multiset1 и Multiset2, причём кратности элементов складываются (т.е. если X содержится 2 раза в MS1 и 3 раза в MS2, то оно содержится в union(MS1, MS2) 5 раз). 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса.
15	1. Реализуйте функцию flatten(bin_tree) -> list. flatten(Tree) возвращает список всех данных в дереве в порядке возрастания. 2. Разработайте интерфейс для абстрактного типа данных "словарь" (см. http://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2). Словарь позволяет хранить произвольное число пар ключ-значение без определённого порядка, при этом две пары с одним ключом одновременно не допускаются. 3. Реализуйте над этим интерфейсом функцию all_values(dictionary) -> [any]. all_values(Dict) возвращает список всех значений в словаре. 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса.
16	1. Реализуйте функцию count_leaves(bin_tree) -> integer. count_leaves(Tree) возвращает число листьев дерева (т.е. вершин, у которых оба поддерева -- пустые).

	2. Разработайте интерфейс для абстрактного типа данных "очередь с приоритетом" (см. http://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C_%D1%81_%D0%BF%D1%80%D0%B8%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82%D0%BE%D0%BC). Очередь с приоритетом позволяет хранить пары (значение, приоритет), при этом каждое значение может храниться несколько раз (в том числе с одинаковым приоритетом) и поддерживает операцию извлечения пары с минимальным приоритетом. 3. Реализуйте над этим интерфейсом функцию to_list(priority_queue) -> [any]. to_list(Queue) возвращает список всех значений, содержащихся в Queue, в порядке возрастания приоритета. 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса.
17	1. Реализуйте функцию split(bin_tree, any) -> {bin_tree, bin_tree}. split(Tree, X) возвращает пару {TreeLT, TreeGT}; TreeLT содержит все элементы Tree, меньшие X, а TreeGT -- все элементы, большие X. 2. Разработайте интерфейс для абстрактного типа данных "множество" (см. http://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE_%28%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%29). Множество позволяет хранить произвольное число значений без определённого порядка, при этом каждое значение хранится не более одного раза. 3. Реализуйте над этим интерфейсом функцию is_subset(set, set) -> bool. is_subset(Set1, Set2) возвращает true, если Set1 -- подмножество Set2 (т.е. все элементы Set1 содержатся в Set2). 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса. Каждая реализация в своём модуле.
18	1. Реализуйте функцию merge(bin_tree, bin_tree) -> bin_tree. merge(Tree1, Tree2) возвращает дерево, содержащее все элементы Tree1 и Tree2. 2. Разработайте интерфейс для абстрактного типа данных "мультимножество" (см. http://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE). Мультимножество позволяет хранить произвольное число значений без определённого порядка, при этом для каждого значение хранится также число раз, которое этот объект входит в мультимножество). 3. Реализуйте над этим интерфейсом функцию union(multiset, multiset) -> bool. union(Multiset1, Multiset2) возвращает мультимножество, содержащее все элементы Multiset1 и Multiset2, причём кратности элементов складываются (т.е. если X содержится 2 раза в MS1 и 3 раза в MS2, то оно содержится в union(MS1, MS2) 5 раз). 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса.
19	1. Реализуйте функцию flatten(bin_tree) -> list. flatten(Tree) возвращает список всех данных в дереве в порядке возрастания.

	2. Разработайте интерфейс для абстрактного типа данных "словарь" (см. http://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2). Словарь позволяет хранить произвольное число пар ключ-значение без определённого порядка, при этом две пары с одним ключом одновременно не допускаются. 3. Реализуйте над этим интерфейсом функцию all_values(dictionary) -> [any]. all_values(Dict) возвращает список всех значений в словаре. 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса.
20	1. Реализуйте функцию count_leaves(bin_tree) -> integer. count_leaves(Tree) возвращает число листьев дерева (т.е. вершин, у которых оба поддерева -- пустые). 2. Разработайте интерфейс для абстрактного типа данных "очередь с приоритетом" (см. http://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C_%D1%81_%D0%BF%D1%80%D0%B8%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82%D0%BE%D0%BC). Очередь с приоритетом позволяет хранить пары (значение, приоритет), при этом каждое значение может храниться несколько раз (в том числе с одинаковым приоритетом) и поддерживает операцию извлечения пары с минимальным приоритетом. 3. Реализуйте над этим интерфейсом функцию to_list(priority_queue) -> [any]. to_list(Queue) возвращает список всех значений, содержащихся в Queue, в порядке возрастания приоритета. 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса.
21	1. Реализуйте функцию split(bin_tree, any) -> {bin_tree, bin_tree}. split(Tree, X) возвращает пару {TreeLT, TreeGT}; TreeLT содержит все элементы Tree, меньшие X, а TreeGT -- все элементы, большие X. 2. Разработайте интерфейс для абстрактного типа данных "множество" (см. http://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE_%28%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%29). Множество позволяет хранить произвольное число значений без определённого порядка, при этом каждое значение хранится не более одного раза. 3. Реализуйте над этим интерфейсом функцию is_subset(set, set) -> bool. is_subset(Set1, Set2) возвращает true, если Set1 -- подмножество Set2 (т.е. все элементы Set1 содержатся в Set2). 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса. Каждая реализация в своём модуле.
22	1. Реализуйте функцию merge(bin_tree, bin_tree) -> bin_tree. merge(Tree1, Tree2) возвращает дерево, содержащее все элементы Tree1 и Tree2. 2. Разработайте интерфейс для абстрактного типа данных "мультимножество" (см. http://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE). Мультимножество позволяет хранить произвольное число значений без

	определённого порядка, при этом для каждого значение хранится также число раз, которое этот объект входит в мультимножество). 3. Реализуйте над этим интерфейсом функцию union(multiset, multiset) -> bool. union(Multiset1, Multiset2) возвращает мультимножество, содержащее все элементы Multiset1 и Multiset2, причём кратности элементов складываются (т.е. если X содержится 2 раза в MS1 и 3 раза в MS2, то оно содержится в union(MS1, MS2) 5 раз). 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса.
23	1. Реализуйте функцию flatten(bin_tree) -> list. flatten(Tree) возвращает список всех данных в дереве в порядке возрастания. 2. Разработайте интерфейс для абстрактного типа данных "словарь" (см. http://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2). Словарь позволяет хранить произвольное число пар ключ-значение без определённого порядка, при этом две пары с одним ключом одновременно не допускаются. 3. Реализуйте над этим интерфейсом функцию all_values(dictionary) -> [any]. all_values(Dict) возвращает список всех значений в словаре. 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса.
24	1. Реализуйте функцию count_leaves(bin_tree) -> integer. count_leaves(Tree) возвращает число листьев дерева (т.е. вершин, у которых оба поддерева -- пустые). 2. Разработайте интерфейс для абстрактного типа данных "очередь с приоритетом" (см. http://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C_%D1%81_%D0%BF%D1%80%D0%B8%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82%D0%BE%D0%BC). Очередь с приоритетом позволяет хранить пары (значение, приоритет), при этом каждое значение может храниться несколько раз (в том числе с одинаковым приоритетом) и поддерживает операцию извлечения пары с минимальным приоритетом. 3. Реализуйте над этим интерфейсом функцию to_list(priority_queue) -> [any]. to_list(Queue) возвращает список всех значений, содержащихся в Queue, в порядке возрастания приоритета. 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса.
25	1. Реализуйте функцию split(bin_tree, any) -> {bin_tree, bin_tree}. split(Tree, X) возвращает пару {TreeLT, TreeGT}; TreeLT содержит все элементы Tree, меньшие X, а TreeGT -- все элементы, большие X. 2. Разработайте интерфейс для абстрактного типа данных "множество" (см. http://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE_%28%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%29). Множество позволяет хранить произвольное число значений без определённого порядка, при этом каждое значение хранится не более одного раза.

	3. Реализуйте над этим интерфейсом функцию is_subset(set, set) -> bool. is_subset(Set1, Set2) возвращает true, если Set1 -- подмножество Set2 (т.е. все элементы Set1 содержатся в Set2). 4. Разработайте 1 (для частичного зачёта) или 2 реализации этого интерфейса. Каждая реализация в своём модуле.

Дополнительные задания:

5. Оцените алгоритмическую сложность реализаций, созданных в задании 4.

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		
Верно выполнено дополнительное задание 5		
Итого	7	3,5