МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
Обнинский институт атомной энергетики –
филиал федерального государственного автономного образовательного учреждения высшего образования «Национальный исследовательский ядерный университет «МИФИ»
(ИАТЭ НИЯУ МИФИ)

Одобрено УМС ИАТЭ НИЯУ МИФИ,
Протокол №2-8/2021 От 30.08.2021

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

«Логическое программирование»

Направление подготовки:	09.03.01 «Информатика и вычислительная техника»
Профиль:	«Вычислительные машины, комплексы, системы и сети»
Квалификация (степень) выпускника:	бакалавр
Форма обучения:	очная

2021 г.

Фонд оценочных средств составлен в соответствии требованиями образовательного стандарта высшего образования национального исследовательского ядерного университета «МИФИ» по направлению подготовки 09.03.01 – Информационные системы и технологии (уровень бакалавриата), приказа Минобрнауки России №1367 от 19.12.2013 г.

Фонд оценочных средств составили:

_____ А.В. Васяшин, ст. преподаватель каф. АСУ

Фонд оценочных средств рассмотрен на заседании отделения интеллектуальных кибернетических систем (О)
(протокол № 5/7 от «30» июля 2021 г.)

Руководитель образовательной программы
09.03.01 Информатика и вычислительная техника
_____ С.О. Старков
«30» июля 2021 г.

Область применения

Фонд оценочных средств (ФОС) – является неотъемлемой частью учебно-методического комплекса учебной дисциплины «Логическое программирование» и предназначен для контроля и оценки образовательных достижений обучающихся, освоивших программу данной дисциплины.

Цели и задачи фонда оценочных средств

Целью Фонда оценочных средств является установление соответствия уровня подготовки обучающихся требованиям федерального государственного образовательного стандарта.

Для достижения поставленной цели Фондом оценочных средств по дисциплине «Логическое программирование» решаются следующие задачи:

- контроль и управление процессом приобретения обучающимися знаний, умений и навыков предусмотренных в рамках данного курса;
- контроль и оценка степени освоения компетенций предусмотренных в рамках данного курса;
- обеспечение соответствия результатов обучения задачам будущей профессиональной деятельности через совершенствование традиционных и внедрение инновационных методов обучения в образовательный процесс в рамках данного курса.

1. Перечень компетенций с указанием этапов их формирования в процессе освоения образовательной программы

1.1. Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы

В результате освоения ООП бакалавриата обучающийся должен овладеть следующими результатами обучения по дисциплине:

Коды компетенций	Результаты освоения ООП *Содержание компетенций*	Перечень планируемых результатов обучения по дисциплине
ПК-3	Способен разрабатывать модели и компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии	Знать: основы и принципы логического программирования, возможности использования логического программирования в решении задач искусственного интеллекта. Уметь: разрабатывать программы и продукционные системы на языке «Пролог», . Владеть: основными приемами и методами разработки программ на языке «Пролог», способами организации дедуктивных баз данных и реализации символических вычислений в языках логического программирования.

1.2. Этапы формирования компетенций в процессе освоения ООП бакалавриата

Компоненты компетенций, как правило, формируются при изучении нескольких дисциплин, а также в немалой степени в процессе прохождения практик, НИР и во время самостоятельной работы обучающегося. Выполнение и защита ВКР являются видом учебной деятельности, который завершает процесс формирования компетенций.

Место дисциплины и соответствующий этап формирования компетенций в целостном процессе подготовки по образовательной программе можно определить по матрице компетенций.

Этапы формирования компетенции в процессе освоения дисциплины:

- **начальный** этап – на этом этапе формируются основы знаний и инструментальные основы компетенции, осваиваются основные категории, формируются базовые умения. Студент воспроизводит термины, факты, методы, понятия, принципы и правила; решает учебные задачи по образцу;
- **основной** этап – знания, умения, навыки, обеспечивающие формирование компетенции, значительно возрастают, но еще не достигают итоговых значений. На этом этапе студент осваивает аналитические действия с предметными знаниями по дисциплине, способен самостоятельно решать учебные задачи, внося коррективы в алгоритм действий, осуществляя коррекцию в ходе работы, переносит знания и умения на новые условия;
- **завершающий** этап – на этом этапе студент достигает итоговых показателей по заявленной компетенции, то есть осваивает весь необходимый объем знаний, овладевает всеми умениями и навыками в сфере заявленной компетенции. Он способен использовать эти знания, умения, навыки при решении задач повышенной сложности и в нестандартных условиях.

Этапы формирования компетенций в ходе освоения дисциплины отражаются в тематическом плане (см.п. 4 рабочей программы дисциплины).

1.3. Паспорт фонда оценочных средств по дисциплине

№ п/п	Контролируемые разделы (темы) дисциплины (результаты по разделам)	Код контролируемой компетенции (или её части) / и ее формулировка	Наименование оценочного средства
Текущий контроль			
1.	Введение в логическое программирование	ПК-3: Способность осваивать методики использования программных средств для решения практических задач	Лабораторные работы 0, 1, Контрольная работа 1
2.	Списки	ПК-3: Способность осваивать методики использования программных средств для решения практических задач	Лабораторные работы 1, 2, Контрольная работа 1
3.	Структуры	ПК-3: Способность осваивать методики использования программных средств для решения практических задач	Лабораторные работы 3, 4, Контрольная работа 1
4.	Интерпретатор и способы управления его работой	ПК-3: Способность осваивать методики использования программных средств для решения практических задач	Лабораторная работа 5, Контрольная работа 2
5.	Организация термов в базы данных	ПК-3: Способность осваивать методики использования программных средств для решения практических задач	Лабораторная работа 5, Контрольная работа 2
6.	Пролог – язык искусственного интеллекта	ПК-3: Способность осваивать методики использования программных средств для решения практических задач	Контрольная работа 2
Промежуточный контроль			
	Зачет по всем темам	ПК-3: Способность осваивать методики использования программных средств для решения практических задач	Отчет по ЛР и две задачи

2. Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания

Конечными результатами освоения программы дисциплины являются сформированные когнитивные дескрипторы «знать», «уметь», «владеть», расписанные по отдельным компетенциям, которые приведены в п.1.1. Формирование этих дескрипторов происходит в процессе изучения дисциплины по этапам в рамках различного вида учебных занятий и самостоятельной работы.

Выделяются три уровня сформированности компетенций на каждом этапе: пороговый, продвинутый и высокий.

Уровни	Содержательное описание уровня	Основные признаки выделения уровня	БРС, % освоения	ECTS/Пятибалльная шкала для оценки экзамена/зачета
Высокий *Все виды компетенций сформированы на высоком уровне в соответствии с целями и задачами дисциплины*	Творческая деятельность	*Включает нижестоящий уровень.* Студент демонстрирует свободное обладание компетенциями, способен применить их в нестандартных ситуациях: показывает умение самостоятельно принимать решение, решать проблему/задачу теоретического или прикладного характера на основе изученных методов, приемов, технологий	90-100	A/ Отлично/ Зачтено
Продвинутый *Все виды компетенций сформированы на продвинутом уровне в соответствии с целями и задачами дисциплины*	Применение знаний и умений в более широких контекстах учебной и профессиональной деятельности, нежели по образцу, большей долей самостоятельности и инициативы	*Включает нижестоящий уровень.* Студент может доказать владение компетенциями: демонстрирует способность собирать, систематизировать, анализировать и грамотно использовать информацию из самостоятельно найденных теоретических источников и иллюстрировать ими теоретические положения или обосновывать практику применения.	85-89	B/ Очень хорошо/ Зачтено
			75-84	C/ Хорошо/ Зачтено
Пороговый *Все виды компетенций сформированы на пороговом уровне*	Репродуктивная деятельность	Студент демонстрирует владение компетенциями в стандартных ситуациях: излагает в пределах задач курса теоретически и практически контролируемый материал.	65-74	D/Удовлетворительно/ Зачтено
			60-64	E/Посредственно /Зачтено
Ниже порогового	Отсутствие признаков порогового уровня: компетенции не сформированы. Студент не в состоянии продемонстрировать обладание компетенциями в стандартных ситуациях.		0-59	Неудовлетворительно/ Зачтено

Оценивание результатов обучения студентов по дисциплине осуществляется по регламенту текущего контроля и промежуточной аттестации.

Критерии оценивания компетенций на каждом этапе изучения дисциплины для каждого вида оценочного средства и приводятся в п. 4 ФОС. Итоговый уровень сформированности компетенции при изучении дисциплины определяется по таблице. При этом следует понимать, что граница между уровнями для конкретных результатов освоения образовательной программы может смещаться.

Уровень сформированности компетенции	**Текущий контроль**	**Промежуточная аттестация**
высокий	**высокий**	**высокий**
	продвинутый	*высокий*
	высокий	*продвинутый*
продвинутый	*пороговый*	*высокий*
	высокий	*пороговый*
	продвинутый	**продвинутый**
	продвинутый	*пороговый*
	пороговый	*продвинутый*
пороговый	**пороговый**	**пороговый**
ниже порогового	**пороговый**	**ниже порогового**
	ниже порогового	**-**

3. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков или опыта деятельности, характеризующих этапы формирования компетенций.

Рейтинговая оценка знаний является интегральным показателем качества теоретических и практических знаний и навыков студентов по дисциплине и складывается из оценок, полученных в ходе текущего контроля и промежуточной аттестации.

Текущий контроль в семестре проводится с целью обеспечения своевременной обратной связи, для коррекции обучения, активизации самостоятельной работы студентов.

Промежуточная аттестация предназначена для объективного подтверждения и оценивания достигнутых результатов обучения после завершения изучения дисциплины.

Текущий контроль осуществляется два раза в семестр: контрольная точка № 1 (КТ № 1) и контрольная точка № 2 (КТ № 2).

Результаты текущего контроля и промежуточной аттестации подводятся по шкале балльно-рейтинговой системы.

Вид контроля	**Этап рейтинговой системы, оценочное средство**	**Балл**	
		Минимум	Максимум
Текущий	**Контрольная точка № 1**	18	30
	Лабораторная работа № 0	2	3
	Лабораторная работа № 1	4	7
	Лабораторная работа № 2	4	7
	Лабораторная работа № 3	4	7
	Контрольная работа № 1	4	6
	Контрольная точка № 2	18	30
	Лабораторная работа № 4	7	12
	Контрольная работа № 2	4	6
	Лабораторная работа № 5	7	12
Промежуточный	**Зачет**	24	40
	Контрольная работа № 1	10	15
	Контрольная работа № 2	10	15
	Оформление отчета по лабораторным работам	4	10
ИТОГО по дисциплине		60	100

Нормативные сроки проведения текущего контроля:

Этап рейтинговой системы, оценочное средство	Неделя семестра
Контрольная точка № 1	10
Лабораторная работа № 0	2
Лабораторная работа № 1	4
Лабораторная работа № 2	6
Лабораторная работа № 3	8
Контрольная работа № 1	10
Контрольная точка № 2	17
Лабораторная работа № 4	11
Контрольная работа № 2	15
Лабораторная работа № 5	17

Контрольные работы проводятся и оцениваются лектором. Прием лабораторных работ осуществляется «с экрана». Лабораторная работа №0 является вводным занятием и заключается в освоении приемов работы в интерпретаторе Пролога. Три балла за эту работу выставляются при выполнении работы в установленные сроки, а два – в противном случае. Все остальные лабораторные выполняются по индивидуальным заданиям и описываются далее в настоящем документе.

4.Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков
4.1. Зачет

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
Обнинский институт атомной энергетики –
филиал федерального государственного автономного образовательного учреждения высшего профессионального образования «Национальный исследовательский ядерный университет «МИФИ»

Кафедра автоматизированных систем управления

Направление **09.03.01 «Информатика и вычислительная техника»**
Программа **«Автоматизированные системы обработки информации и управления»**
«Электронно-вычислительные машины (ЭВМ), системы и сети»
Дисциплина **«Логическое программирование»**

Зачет с оценкой выставляется по результатам выполнения лабораторных работ, оформленных в виде отчета, при условии выполнения всех лабораторных работ и решения двух задач (см. описания контрольных работ).

Оценка отчета по лабораторным работам

Оценка (баллы)	**Критерии оценки**
Отлично 9–10	Наличие всех необходимых структурных элементов отчета, полное исчерпывающее изложение результатов работы, изложение грамотным четким и ясным языком, соблюдение правил оформления
Хорошо 7–8	Наличие всех необходимых структурных элементов отчета, полное изложение результатов работы, наличие незначительного числа опечаток, синтаксических ошибок и погрешностей в стиле изложения, незначительные нарушения правил оформления
Удовлетворительно 4–6	Наличие всех необходимых структурных элементов отчета, полное изложение результатов работы, наличие опечаток, синтаксических ошибок и погрешностей в стиле изложения, нарушение правил оформления
Неудовлетворительно 0–3	Отсутствие всех необходимых структурных элементов отчета, неполное изложение результатов работы, наличие большого числа опечаток, синтаксических ошибок, слабый стиль изложения, грубые нарушения правил оформления

Оценка за решение задачи на зачете

Оценка (баллы)	**Критерии оценки**
Отлично 15	Задача решена полностью правильно.
Хорошо 14	В описании решения задачи есть ошибки, которые можно трактовать как описки, например, из-за невнимательности.
Удовлетворительно 10-13	Алгоритм решения задачи выбран правильно, но в ряде мест сделаны ошибки.
Неудовлетворительно 0–9	Неверен алгоритм решения задачи.

Общая оценка за зачет

Оценка	**Критерии оценки**
Отлично 38–40	Складывается из оценок, полученных за лабораторные работы и результатам решения двух задач и активность при условии, что все эти оценки положительные (удовлетворительно, хорошо или отлично)
Хорошо 34–37	
Удовлетворительно 24–33	
Неудовлетворительно 0–23	Оценка «неудовлетворительно» ставится, если хотя бы одна оценка за лабораторные работы, или задачу – «неудовлетворительно». Численное значение оценки равно сумме полученных баллов. Если сумма превышает 23, то ставится 23.

4.2. Контрольная работа № 1

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
Обнинский институт атомной энергетики –
филиал федерального государственного автономного образовательного учреждения высшего профессионального образования «Национальный исследовательский ядерный университет «МИФИ»

Кафедра автоматизированных систем управления

Комплект заданий для контрольной работы

по дисциплине Логическое программирование

Темы: «Введение в логическое программирование», «Списки», «Структуры»

Контрольная работа представляет собой практическую задачу, на который студент должен дать исчерпывающий письменный ответ в виде кода процедуры на языке Пролог. Перечень вариантов:

Задача №1

Разработать предикат, который по исходному списку (первый аргумент), возвращает список (второй аргумент), полученный из исходного циклическим сдвигом влево на случайное число позиций.

Задача №2

Разработать предикат, который по исходному списку (первый аргумент), возвращает список (второй аргумент), полученный из исходного удалением случайного числа элементов исходного списка. Вероятность вхождения элемента исходного списка в результат, должна быть равна вероятности его отсутствия в результирующем списке.

Задача №3

Разработать предикат, который по двум исходным, упорядоченным по возрастанию спискам (первый и второй аргументы), возвращает список (третий аргумент), являющийся объединением исходных списков, который также должен быть упорядочен по возрастанию. (Примечание: запрещается использовать встроенный предикат **sort**)

Задача №4

Разработать предикат, который по двум исходным спискам (первый и второй аргументы), определяет, являются ли списки одинаковыми (с точностью до перестановки элементов). Предполагается, что каждый из исходных списков может иметь повторяющиеся элементы.

Задача №5

Разработать предикат, который по исходному бинарному дереву (первый аргумент), возвращает список (второй аргумент), содержащий листья исходного дерева.

Задача №6

Разработать предикат, который по исходному бинарному дереву (первый аргумент), возвращает поддерево (второй аргумент) исходного дерева. При отказе от ответа, предикат должен возвращать другое поддерево и т.д. пока не будут возвращены все поддеревья.

Задача №7

Разработать предикат, который по исходному бинарному дереву (первый аргумент), возвращает дерево (второй аргумент), получающееся из исходного дерева, удалением одного из его поддеревьев. При отказе от ответа, предикат должен возвращать другое дерево и т.д. пока не будут возвращены все деревья, которые можно получить из исходного.

Задача №8

Разработать предикат, который по двум исходным бинарным деревьям (первый и второй аргументы), определяет, являются ли они эквивалентными.

Задача №9

Разработать предикат, который по двум исходным бинарным деревьям (первый и второй аргументы), определяет, является ли первое из них поддеревом второго.

Задача №10

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит дерево (второй аргумент), путем перестановки поддеревьев. Перестановка производится только тогда, когда правым поддеревом является лист дерева, а левое поддерево листом не является.

Задача №11

Разработать предикат, который по двум исходным бинарным деревьям (первый и второй аргументы), определяет, является ли первое из них частичным поддеревом второго.

Задача №12

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит список (второй аргумент), содержащий наименования вершин лежащих на пути от корня дерева до листа. В ходе последовательного отказа от ответа предикат должен возвращать все такие пути.

Задача №13

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит список (третий аргумент), содержащий наименования вершин лежащих на глубине, заданной вторым аргументом.

Задача №14

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит список (второй аргумент), содержащий длины всех путей от корня дерева до его листьев.

Задача №15

Разработать предикат, который по исходному, произвольному дереву (первый аргумент), возвращает поддерево (второй аргумент) исходного дерева. При отказе от ответа, предикат должен возвращать другое поддерево и т.д. пока не будут возвращены все поддеревья.

Задача №16

Разработать предикат, который по исходному списку (первый аргумент), возвращает список (второй аргумент), полученный из исходного циклическим сдвигом вправо на случайное число позиций.

Задача №17

Разработать предикат, который по исходному списку (первый аргумент), возвращает список (второй аргумент), полученный из исходного дублированием каждого элемента исходного списка.

Задача №18

Разработать предикат, который по двум исходным, упорядоченным по убыванию спискам (первый и второй аргументы), возвращает список (третий аргумент), являющийся объединением исходных списков, который также должен быть упорядочен по убыванию. (Примечание: запрещается использовать встроенный предикат **sort**)

Задача №19

Разработать предикат, который по исходному бинарному дереву (первый аргумент), возвращает список (второй аргумент), содержащий такие узлы, у которых среди дочерних узлов есть листья исходного дерева.

Задача №20

Разработать предикат, который по исходному бинарному дереву (первый аргумент), возвращает дерево (второй аргумент), получающееся из исходного дерева, дублированием одного из его поддеревьев. При отказе от ответа, предикат должен возвращать другое дерево и т.д. пока не будут возвращены все деревья, которые можно получить из исходного.

Задача №21

Разработать предикат, который по двум исходным бинарным деревьям (первый и второй аргументы), определяет, являются ли они эквивалентными.

Задача №22

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит дерево (второй аргумент), путем перестановки листьев (если оба дочерних узла дерева - листья).

Задача №23

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит список (второй аргумент), содержащий длины путей от корня дерева до его листьев.

Задача №24

Разработать предикат, который по исходному списку (первый аргумент), возвращает список (второй аргумент), полученный из исходного циклическим сдвигом влево на случайное число позиций.

Задача №25

Разработать предикат, который по исходному списку (первый аргумент), возвращает список (второй аргумент), полученный из исходного удалением случайного числа элементов исходного списка. Вероятность вхождения элемента исходного списка в результат, должна быть равна вероятности его отсутствия в результирующем списке.

Задача №26

Разработать предикат, который по двум исходным, упорядоченным по возрастанию спискам (первый и второй аргументы), возвращает список (третий аргумент), являющийся объединением исходных списков, который также должен быть упорядочен по возрастанию. (Примечание: запрещается использовать встроенный предикат **sort**)

Задача №27

Разработать предикат, который по исходному бинарному дереву (первый аргумент), возвращает список (второй аргумент), содержащий листья исходного дерева.

Задача №28

Разработать предикат, который по исходному бинарному дереву (первый аргумент), возвращает поддерево (второй аргумент) исходного дерева. При отказе от ответа, предикат должен возвращать другое поддерево и т.д. пока не будут возвращены все поддеревья.

Задача №29

Разработать предикат, который по исходному бинарному дереву (первый аргумент), возвращает дерево (второй аргумент), получающееся из исходного дерева, удалением одного из его поддеревьев. При отказе от ответа, предикат должен возвращать другое дерево и т.д. пока не будут возвращены все деревья, которые можно получить из исходного.

Задача №30

Разработать предикат, который по двум исходным бинарным деревьям (первый и второй аргументы), определяет, являются ли они эквивалентными.

Задача №31

Разработать предикат, который по двум исходным бинарным деревьям (первый и второй аргументы), определяет, является ли первое из них поддеревом второго.

Задача №32

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит дерево (второй аргумент), путем перестановки поддеревьев. Перестановка производится только тогда, когда правым поддеревом является лист дерева, а левое поддерево листом не является.

Задача №33

Разработать предикат, который по двум исходным бинарным деревьям (первый и второй аргументы), определяет, является ли первое из них частичным поддеревом второго.

Задача №34

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит список (второй аргумент), содержащий наименования вершин лежащих на пути от корня дерева до листа. В ходе последовательного отказа от ответа предикат должен возвращать все такие пути.

Задача №35

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит список (третий аргумент), содержащий наименования вершин лежащих на глубине, заданной вторым аргументом.

Задача №36

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит список (второй аргумент), содержащий длины всех путей от корня дерева до его листьев.

Задача №37

Разработать предикат, который по исходному, произвольному дереву (первый аргумент), возвращает поддерево (второй аргумент) исходного дерева. При отказе от ответа, предикат должен возвращать другое поддерево и т.д. пока не будут возвращены все поддеревья.

Задача №38

Разработать предикат, который по исходному списку (первый аргумент), возвращает список (второй аргумент), полученный из исходного циклическим сдвигом вправо на случайное число позиций.

Задача №39

Разработать предикат, который по исходному списку (первый аргумент), возвращает список (второй аргумент), полученный из исходного дублированием каждого элемента исходного списка.

Задача №40

Разработать предикат, который по двум исходным, упорядоченным по убыванию спискам (первый и второй аргументы), возвращает список (третий аргумент), являющийся объединением исходных списков, который также должен быть упорядочен по убыванию. (Примечание: запрещается использовать встроенный предикат **sort**)

Задача №41

Разработать предикат, который по исходному бинарному дереву (первый аргумент), возвращает список (второй аргумент), содержащий такие узлы, у которых среди дочерних узлов есть листья исходного дерева.

Задача №42

Разработать предикат, который по исходному бинарному дереву (первый аргумент), возвращает дерево (второй аргумент), получающееся из исходного дерева, дублированием одного из его поддеревьев. При отказе от ответа, предикат должен возвращать другое дерево и т.д. пока не будут возвращены все деревья, которые можно получить из исходного.

Задача №43

Разработать предикат, который по двум исходным бинарным деревьям (первый и второй аргументы), определяет, являются ли они эквивалентными.

Задача №44

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит дерево (второй аргумент), путем перестановки листьев (если оба дочерних узла дерева - листья).

Задача №45

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит список (второй аргумент), содержащий длины путей от корня дерева до его листьев.

Описание шкалы оценивания:

Оценка (баллы)	**Критерии оценки**
Отлично 6	Разработан правильный алгоритм решения, и его реализация в виде процедуры Пролога выполнена без ошибок.
Хорошо 5	Разработан правильный алгоритм решения, но его реализация в виде процедуры Пролога выполнена с небольшими ошибками.
Удовлетворительно 4	Разработан правильный алгоритм решения, но его реализация в виде процедуры Пролога выполнена с существенными ошибками.
Неудовлетворительно 0–3	Алгоритм решения задачи неверен

4.5. Контрольная работа № 2

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
Обнинский институт атомной энергетики –
филиал федерального государственного автономного образовательного учреждения высшего профессионального образования «Национальный исследовательский ядерный университет «МИФИ»

Кафедра автоматизированных систем управления

Комплект заданий для контрольной работы

по дисциплине Логическое программирование

Темы: «Интерпретатор и способы управления его работой», «Организация термов в базы данных», «Пролог – язык искусственного интеллекта»

В данной контрольной работе проверяются знания и навыки работы с термами базы данных и использования одного из основных приёмов программирования – «вынуждаемый возврат». В каждом задании необходимо:

1. Объяснить, что делает предложенный предикат;
2. Указать, какие аргументы предиката должны быть конкретизированы при запросе к нему;
3. Привести все возможные варианты (по конкретизации аргументов) примеров запросов к предикату и раскрыть результаты таких запросов;
4. Объяснить работу интерпретатора по выполнению одного из возможных запросов;
5. Если в качестве подцели в теле одной или нескольких фраз предиката выступает предикат «сократить», пояснить смысл его использования.

Варианты заданий.

Вариант №1 f(Key):- recorded(Key,T,U), recorda(Key,T,U), not(nref(U,_)), !.	Вариант №2 g(Key1,Key2):- recorded(Key1,T,_), not(recorded(Key2,T,_)), !.
Вариант №3 g(Key):- recorded(Key,T,R), nref(R,N), instance(N,F), F < T.	Вариант №4 f(Key1,Key2,T):- recorded(Key1,T,_), recorded(Key2,T,_).
Вариант №5 f_replace(Key):- recorded(Key,Term,R), Term=..L, replace(R,L), fail.	Вариант №6 f_assert(Key):- recorded(Key,Term,_), assert(Term), fail. f_assert(Key):- eraseall(Key).

Вариант №7 c_assert(Key):- recorded(Key,Term,_), (assert(Term) ; retract(Term)).	Вариант №8 d(Key):- recorded(Key,Term,R1), recorded(Key,Term,R2), R1 \== R2, !.
Вариант №9 g(Key1,Key2):- recorded(Key1,T,R), not(recorded(Key2,T,_)), recordz(Key2,T,_), fail.	Вариант №10 r(Key,Term):- recorded(Key,Term,R), erase(R), !. dr(Key,Term):- repeat, not(r(Key,Term)), !, recordz(Key,Term,_).
Вариант №11 c(Key,Term,N):- findall(1, recorded(Key,Term,_), L), length(L,N).	Вариант №12 d(Key):- ctr_set(0,1), recorded(Key,T,R), ctr_inc(0,N), S=..[N,T], replace(R,S), fail.
Вариант №13 g(Key,Term):- recorded(Key,Term,R), nref(R,N), instance(N,Term).	Вариант №14 a(_):- ctr_set(0,0), recorded(f,_,R), ctr_inc(0,_), fail. a(X):- ctr_is(0,X).
Вариант №15 h(Key):- recorded(Key,T,U), nref(U,R), instance(R,G), recorda(Key,G,_), erase(R), fail.	Вариант №16 g(Key,Ref):- recordz(Key,$$,R), pref(R,Ref), erase(Ref).
Вариант №17 n(Key,Term,N):- ctr_set(0,1), repeat, ctr_inc(0,X), nth_ref(Key,X,R), instance(R,T), T = Term, N = X.	Вариант №18 g(Key):- recorded(Key,T,U), nref(U,R), instance(R,G), recorda(Key,G,_), fail.

Вариант №19 n_recorded(Key,Term,Ref):- repeat, recorded(Key,Term,Ref).	Вариант №20 b(X,Y):- ctr_set(0,X), repeat, ctr_is(0,N), Y is N*(N+1), ctr_set(0,Y).
Вариант №21 n_recordz(Key,Term.Ref):- recordz(Key,Term,Ref), (true; (erase(Ref), fail)).	Вариант №22 f(Key):- recorded(Key,T,Ref), !, repeat, ifthenelse(nref(Ref,N), (instance(N,Z), erase(N), recorda(Key,Z,_), fail), true), !.
Вариант №23 r(Key):- recorded(Key,T,R), nref(R,N), erase(N), fail.	Вариант №24 r(Key):- recorded(key,T,R), pref(R,N), erase(N), fail.
Вариант №25 r(Key):- recorded(key,T,R), pref(R,N1), pref(N1,N), erase(N), fail.	Вариант №26 c(Key):- recorded(Key,T,R), erase(R), recordz(Key,T,_), !. dc(Key):- repeat, c(Key).
Вариант №27 f(Key):- recorded(Key,T,R), pref(R,P), instance(P,V), ifthen(V > T, erase(P)), fail.	Вариант №28 g(Key1,Key2):- recorded(Key1,_,Ref), nref(Ref,N), instance(N,T), erase(N), recordz(Key2,T,_), fail.

Вариант №29 h(Key):- recorded(key,T,Ref), nref(Ref,N), instance(N,V), replace(Ref,(T,V)), erase(N), fail.	

Описание шкалы оценивания:

Оценка (баллы)	**Критерии оценки**
Отлично 6	Все пять пунктов задания выполнено
Хорошо 5	Первые три пункта задания выполнены правильно
Удовлетворительно 4	Первые два пункта задания выполнены правильно
Неудовлетворительно 0–3	Один или оба из первых двух пунктов задания выполнены неправильно

4.6. Лабораторная работа № 1

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
Обнинский институт атомной энергетики –
филиал федерального государственного автономного образовательного учреждения высшего профессионального образования «Национальный исследовательский ядерный университет «МИФИ»

Кафедра автоматизированных систем управления

Комплект заданий к лабораторной работе

по дисциплине Логическое программирование

Темы: «Списки»

1. Определить, является ли список упорядоченным по возрастанию.

Аргументы: *исходный список*.

```
?- pred([a,b,d,f]).
yes
?- pred([[a,f,b,d]).
no
?-
```

2. Определить, имеет ли список четное число элементов, не прибегая к подсчету длины списка.

Аргументы: *исходный список*.

```
?- pred([a,b,d,f]).
yes
?- pred([[a,b,d]).
no
?-
```

3. Перевести число из десятичной системы счисления в двоичную.

Аргументы: *число в десятичной с.с*.;
число в двоичной с.с. (*список*).

```
?- pred(26,X).
X = [1,1,0,1,0]
yes
?-
```

4. Найти в неупорядоченном списке вещественных чисел ближайших соседей – два числа, расстояние между которыми на вещественной оси является минимальным среди всех возможных пар.

Аргументы: *исходный список*;
ближайшие соседи (*список из двух чисел*).

```
?- pred([0.1,0.04,8.6,7.56,8.11],X).
X = [8.6,8.11]
yes
?-
```

5. Перевести число из двоичной системы счисления в десятичную.

Аргументы: *число в двоичной с.с*. (*список*);
число в десятичной с.с.

```
?- pred([1,1,0,1,0],X).
X = 26
yes
?-
```

6. Перевести число из десятичной системы счисления в римскую.
Аргументы: *число в десятичной с.с.*;
число в римской с.с. (*список*).

```
?- pred(24,X).
X = [x,x,i,v]
yes
?-
```

7. Перевести число из римской системы счисления в десятичную.
Аргументы: *число в римской с.с.* (*список*);
число в десятичной с.с.

```
?- pred([x,x,i,v],X).
X = 24
yes
?-
```

8. Перевести число из двоичной системы счисления в шестнадцатеричную.
Аргументы: *число в двоичной с.с.* (*список*);
число в шестнадцатеричной с.с. (*список*).

```
?- pred([1,1,0,1,0],X).
X = [1,a]
yes
?-
```

9. Перевести число из шестнадцатеричной системы счисления в двоичную.
Аргументы: *число в шестнадцатеричной с.с.* (*список*);
число в двоичной с.с. (*список*).

```
?- pred([1,a],X).
X = [1,1,0,1,0]
yes
?-
```

10. Определить, является ли первый список подмножеством второго. Списки не имеют дубликатов.
Аргументы: *первый список*;
второй список.

```
?- pred([a,b,c],[s,a,e,c,d,b]).
yes
?-
```

11. Удалить из списка элементы с четными номерами.
Аргументы: *исходный список*;
результирующий список.

```
?- pred([a,b,c,d,e],X).
X = [a,c,e]
yes
?-
```

12. Удалить из списка те элементы, которые не имеют дубликатов.
Аргументы: *исходный список*;
результирующий список.

```
?- pred([a,e,c,d,e],X).
X = [e,e]
yes
?-
```

13. Дан числовой список – [*A*,*B*,*C*,*D*,*E*,…]. Посчитать результат по формуле $R = A + B - C + D - E + \ldots$.
Аргументы: *исходный список*;
результат вычислений.

```
?- pred([1,2,3,4,5],X).
X = -1
yes
?-
```

14. Сформировать по заданному мужскому имени отчество.
Аргументы: *исходная строка* (*имя*);
результирующая строка (*отчество*).

```
?- pred('Иван',X).
X = Иванович
yes
?-
```

15. Дан список, содержащий списки одинаковой длины (матрица). Построить транспонированную матрицу.
Аргументы: *матрица* (*список*);
транспонированная матрица (*список*).

```
?- pred([[1,2,3],[4,5,6],[7,8,9]],X).
X = [[1,4,7],[2,5,8],[3,6,9]]
yes
?-
```

16. Дан список, содержащий списки одинаковой длины (квадратная матрица). Посчитать детерминант.
Аргументы: *матрица* (*список*);
детерминант.

```
?- pred([[1,2],[4,5]],X).
X = -3
yes
?-
```

17. Удалить из числового списка элементы, меньшие заданного числа.
Аргументы: *исходный список*;
число;
результирующий список.

```
?- pred([3,1,5,4,8],5,X).
X = [5,8]
yes
?-
```

18. Удалить из списка элемент с заданным порядковым номером.
Аргументы: *исходный список*;
порядковый номер;
результирующий список.

```
?- pred([a,b,c,d,e],4,X).
X = [a,b,c,e]
yes
?-
```

19. Поместить в результирующий список каждый элемент исходного списка с заданной вероятностью.
Аргументы: *исходный список*;
вероятность вхождения;
результирующий список.

```
?- pred([a,b,c,d,e],0.5,X).

X = [a,c,d]
yes
?- pred([a,b,c,d,e],0.5,X).
X = [b,c,d,e]
yes
?-
```

20. Выполнить циклический сдвиг списка влево на заданное количество элементов.
Аргументы: *исходный список*;
количество элементов;
результирующий список.

```
?- pred([a,b,c,d,e],3,X).
X = [d,e,a,b,c]
yes
?-
```

21. Выполнить циклический сдвиг списка вправо на заданное количество элементов.
Аргументы: *исходный список*;
количество элементов;
результирующий список.

```
?- pred([a,b,c,d,e],3,X).
X = [c,d,e,a,b] -> ;
yes
?-
```

Описание шкалы оценивания:

Оценка (баллы)	**Критерии оценки**
Отлично 7	Лабораторная работа выполнена в срок. Собеседование по лабораторной работе показало владение теоретическим материалом и свободное использование терминологии Пролога при объяснении работы разработанной процедуры.
Хорошо 6	Лабораторная работа выполнена с небольшой задержкой (не более недели). Собеседование по лабораторной работе показало владение теоретическим материалом и свободное использование терминологии Пролога при объяснении работы разработанной процедуры.
Удовлетворительно 4	Лабораторная работа выполнена с большой задержкой (более недели). Собеседование по лабораторной работе показало владение теоретическим материалом и свободное использование терминологии Пролога при объяснении работы разработанной процедуры.
Удовлетворительно 5	Собеседование по лабораторной работе показало слабое владение теоретическим материалом и затруднения в использовании терминологии Пролога при объяснении работы разработанной процедуры.
Неудовлетворительно 0–3	Либо лабораторная работа не выполнена, либо обнаружено полное непонимание разработанной процедуры.

4.7. Лабораторная работа № 2

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
Обнинский институт атомной энергетики –
филиал федерального государственного автономного образовательного учреждения высшего профессионального образования «Национальный исследовательский ядерный университет «МИФИ»

Кафедра автоматизированных систем управления

Комплект заданий к лабораторной работе

по дисциплине Логическое программирование

Темы: «Списки»

1. Получить целое число из строки, использующей числительные русского языка.

Аргументы: *строка*;
целое число.

```
?- pred('Пять тысяч восемьсот одиннадцать',X).
X = 5811
yes
?-
```

2. Даны два списка. Каждый из списков не имеет дубликатов и, следовательно, их можно рассматривать как множества. Построить симметрическую разность (объединение минус пересечение) списков.

Аргументы: *первый список*;
второй список;
результирующий список.

```
?- pred([a,b,c,d,e],[e,w,q,c],X).
X = [a,b,d,w,q] -> ;
yes
?-
```

3. Дан список, содержащий числовые списки одинаковой длины (матрица), и числовой список (вектор). Перемножить матрицу и вектор.

Аргументы: *матрица* (*список*);
вектор (*список*);
результат перемножения (*список*).

```
?- pred([[1,2],[4,5]],[2,3],X).
X = [8,23]
yes
?-
```

4. Дан числовой список [*a*,*b*,*c*,*d*,...] и число *x*. Посчитать значение полинома $R = a+bx+cx^2+dx^3+\dots$.

Аргументы: *список*;
число;
результат вычислений.

```
?- pred([1,2,4,1],3,X).
X = 70
yes
?-
```

5. Преобразовать целое число в строку с использованием числительных русского языка.
Аргументы: *целое число*;
строка.

```
?- pred(5811,X).
X = Пять тысяч восемьсот одиннадцать
yes
?-
```

6. Выполнить циклический сдвиг списка влево.
Аргументы: *исходный список*;
результирующий список.

```
?- pred([a,b,c,d,e],X).
X = [b,c,d,e,a] -> ;
X = [c,d,e,a,b] -> ;
X = [d,e,a,b,c] -> ;
X = [e,a,b,c,d] -> ;
X = [a,b,c,d,e] -> ;
X = [b,c,d,e,a] -> ;
…
…
?-
```

7. Выполнить циклический сдвиг списка вправо.
Аргументы: *исходный список*;
результирующий список.

```
?- pred([a,b,c,d,e],X).
X = [e,a,b,c,d] -> ;
X = [d,e,a,b,c] -> ;
X = [c,d,e,a,b] -> ;
X = [b,c,d,e,a] -> ;
X = [a,b,c,d,e] -> ;
X = [e,a,b,c,d] -> ;
…
…
?-
```

8. Найти порядковый номер максимального элемента числового списка.
Аргументы: *исходный список*;
порядковый номер.

```
?- pred([3,5,8,1,4],X).
X = 3
yes
?-
```

9. Найти все возможные пары элементов списка.
Аргументы: *исходный список*;
пара элементов (*список*).

```
?- pred([a,b,c,d,e],X).
X = [a,b] -> ;
X = [a,c] -> ;
X = [a,d] -> ;
X = [a,e] -> ;
X = [b,c] -> ;
X = [b,d] -> ;
X = [b,e] -> ;
X = [c,d] -> ;
X = [c,e] -> ;
X = [d,e] -> ;
no
?-
```

10. Найти все возможные перестановки элементов списка.
Аргументы: *исходный список*;
результирующий список.

```
?- pred([a,b,c],X).
X = [a,b,c] -> ;
X = [a,c,b] -> ;
X = [b,a,c] -> ;
X = [b,c,a] -> ;
X = [c,a,b] -> ;
X = [c,b,a] -> ;
no
?-
```

11. Строка текста на русском языке содержит «лишние» пробелы. Удалить избыточные пробелы.
Аргументы: *исходная строка*;
результирующая строка.

```
?- pred('Мы  будем рады узнать ваше   мнение!',X).
X = 'Мы будем рады узнать ваше мнение!'
yes
?-
```

12. Даны два числовых списка, содержащие коэффициенты двух полиномов. Соответствие между элементами списка и коэффициентами полинома можно отобразить следующим образом: $a_0+a_1x+a_2x^2+a_3x^3+a_4x^4+\ldots \rightarrow [a_0,a_1,a_2,a_3,a_4,\ldots]$. Перемножить полиномы. Результирующий полином должен представлять собой список коэффициентов, составленный по тому же правилу, что и списки для исходных полиномов.
Аргументы: *первый полином* (*список*);
второй полином (*список*);
результирующий полином.

```
?- pred([3,5,8],[1,4],X).
X = [3,17,28,32]
yes
?-
```

13. Дан список и число *N*. Построить сочетания по *N* элементов из исходного списка.
Аргументы: *исходный список*;
целое число;
результирующий список.

```
?- pred([3,5,8,1],3,X).
X = [3,5,8]
X = [3,5,1]
X = [3,8,1]
X = [5,8,1]
no
?-
```

14. Преобразовать предложение на русском языке в список слов.
Аргументы: *исходная строка*;
список слов (*строк*).

```
?- pred('Это будет работать, но возникнут две проблемы.',X).
X = [Это,будет,работать,но,возникнут,две,проблемы]
yes
?-
```

15. Восстановить по отчеству имя.
Аргументы: *исходная строка* (*отчество*);
результирующая строка (*имя*).

```
?- pred('Иванович',X).
X = Иван
yes
?-
```

16. Подсчитать количество вхождений каждой буквы в строку текста на русском языке и собрать результаты в список.

Аргументы: *исходная строка*;
список пар вида «Буква : *целое число»*.

```
?- pred('Это будет работать',X).
X = [Э:1,т:4,о:2,б:2,у:1,д:1,е:1,р:1,а:2,ь:1]
yes
?-
```

17. Произвести перекодировку строки текста на русском языке из кодировки DOS в кодировку Windows.

Аргументы: *исходная строка* (*DOS*);
результирующая строка (*Windows*).

```
?- pred('Перевод получен',X).
X = ??а?ўRп ЇR<гэ?-
yes
?-
```

18. Дан числовой список, содержащий первые *N* членов ряда, и порядковый номер элемента ряда *K*. Рассчитать *K*-ый элемент ряда при условии, что каждый член ряда равен сумме *N* предыдущих его членов (за исключением *N* первых членов).

Аргументы: *список первых членов ряда*;
порядковый номер члена ряда;
член ряда.

```
?- pred([0,1,2],4,X).
X = 3
yes
?-
```

19. Произвести замену всех вхождений заданной подстроки в исходную строку на новую подстроку.

Аргументы: *исходная строка*;
подстрока для поиска;
подстрока для замены;
результирующая строка.

```
?- pred('голод, холод', 'од', 'одом',X).
X = 'голодом, холодом'
yes
?-
```

20. Дано предложение на русском языке. Построить список слов предложения. Список не должен содержать повторов слов и знаков препинания.

Аргументы: *исходная строка*;
список слов.

```
?- pred('Что воля, что неволя...',X).
X = ['что','воля','неволя']
yes
?-
```

21. Дан список точек плоскости. Каждая точка задается парой координат (X,Y) и, таким образом, исходный список имеет вид [(X_1,Y_1), (X_2,Y_2), (X_3,Y_3),...]. Преобразовать исходный список в список относительных координат вида [(X_1,Y_1), (ΔX_2,ΔY_2), (ΔX_3,ΔY_3),...], где первая точка задается абсолютными координатами, а все последующие – в виде пары (сдвиг по оси X, сдвиг по оси Y) относительно предыдущей точки ($\Delta X_k = X_k - X_{k-1}$, $\Delta Y_k = Y_k - Y_{k-1}$).

Аргументы: *исходный список абсолютных координат*;
список относительных координат.

```
?- pred([(1,2),(3,5),(2,0),(0,0)],X).
X = [(1,2),(2,3),(1,-5),(-2,0)]
yes
?-
```

Описание шкалы оценивания:

Оценка (баллы)	**Критерии оценки**
Отлично 7	Лабораторная работа выполнена в срок. Собеседование по лабораторной работе показало владение теоретическим материалом и свободное использование терминологии Пролога при объяснении работы разработанной процедуры.
Хорошо 6	Лабораторная работа выполнена с небольшой задержкой (не более недели). Собеседование по лабораторной работе показало владение теоретическим материалом и свободное использование терминологии Пролога при объяснении работы разработанной процедуры.
Удовлетворительно 4	Лабораторная работа выполнена с большой задержкой (более недели). Собеседование по лабораторной работе показало владение теоретическим материалом и свободное использование терминологии Пролога при объяснении работы разработанной процедуры.
Удовлетворительно 5	Собеседование по лабораторной работе показало слабое владение теоретическим материалом и затруднения в использовании терминологии Пролога при объяснении работы разработанной процедуры.
Неудовлетворительно 0–3	Либо лабораторная работа не выполнена, либо обнаружено полное непонимание разработанной процедуры.

4.8. Лабораторная работа № 3

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
Обнинский институт атомной энергетики –
филиал федерального государственного автономного образовательного учреждения высшего профессионального образования «Национальный исследовательский ядерный университет «МИФИ»

Кафедра автоматизированных систем управления

Комплект заданий к лабораторной работе

по дисциплине Логическое программирование

Темы: «Структуры»

1. Определить наличие на каком-либо из путей от корня до листа хотя бы двух узлов с одинаковым именем.

Аргументы: *произвольное бинарное дерево*.

```
?- pred(a(f(a(m,k),r),n)).
yes
?- pred(a(f(d(m,k),r),n)).
no
?-
```

2. Определить наличие хотя бы двух узлов с одинаковым именем на одной глубине.

Аргументы: *произвольное бинарное дерево*.

```
?- pred(s(f(b(m,k),a),n(b,g))).
yes
?- pred(s(f(b(m,k),a),n(t,g))).
no
?-
```

3. Определить наличие двух одинаковых путей от корня до листа.

Аргументы: *произвольное бинарное дерево*.

```
?- pred(s(f(b(m,k),a),f(a,g))).
yes
?- pred(s(f(y(m,k),a),f(t,g))).
no
?-
```

4. Расширить синтаксис языка Пролог, введя операции (см. встроенный предикат `op/3`) над комплексными числами. Следует использовать имя «`isc`» для операции извлечения результата и стандартные «`+, -, *, /, ^`» для арифметических операций.

Аргументы: *арифметическое выражение*;
свободная переменная.

```
?- X isc (1.6, 4.5) + (2.8, 7.1).
X = (4.4, 11.6)
yes
?-
```

5. Определить глубину дерева.

Аргументы: *произвольное бинарное дерево*;
глубина дерева.

```
?- pred(s(f(b(m,k),a),f(a,g)),X).
X = 4
yes
?-
```

6. Определить, являются ли два заданных дерева равными с точностью до перестановки левого и правого поддерева в каждом узле.

Аргументы: *первое дерево*;
второе дерево.

```
?- pred(s(t(b(m,k),a),f(a,g)), s(t(b(k,m),a),f(g,a))).
yes
?- pred(s(t(b(m,k),a),f(a,g)), s(t(a,b(k,m)),f(g,a))).
yes
?-
```

7. Собрать все узлы в список.

Аргументы: *произвольное бинарное дерево*;
список узлов.

```
?- pred(s(f(b(m,k),a),f(a,g)),X).
X = [s,f,b,m,k,a,f,a,g]
yes
?-
```

8. В каждом узле дерева поменять при необходимости поддеревья местами так, чтобы в результирующем дереве в каждом узле выполнялось требование – количество узлов в левом поддереве не больше, чем в правом.

Аргументы: *произвольное бинарное дерево*;
результирующее дерево.

```
?- pred(s(f(b(m,k),a),t(a,g)),X).
X = s(t(a,g),f(a,b(m,k)))
yes
?-
```

9. Найти максимум количества узлов лежащих на одной глубине.

Аргументы: *произвольное бинарное дерево*;
количество узлов.

```
?- pred(s(f(b(m,k),a),t(r,w)),X).
X = 4
yes
?-
```

10. Собрать в список имена всех узлов дерева, лежащих на заданной глубине.

Аргументы: *произвольное бинарное дерево*;
необходимая глубина;
список узлов.

```
?- pred(s(f(b(m,k),a),t(a,g)),1,X).
X = [s]
yes
?- pred(s(f(b(m,k),a),t(a,g)),3,X).
X = [b,a,a,g]
yes
?- pred(s(f(b(m,k),a),t(a,g)),5,X).
X = []
yes
?-
```

11. Обрезать дерево на заданной глубине.

Аргументы: *произвольное бинарное дерево*;
необходимая глубина;
результирующее дерево.

```
?- pred(s(f(b(m,k),a),t(a,g)),2,X).
X = s(f,t)
yes
?- pred(s(f(b(m,k),a),t(a,g)),3,X).
X = s(f(b,a),t(a,g))
yes
?-
```

12. Произвести следующую процедуру над деревом, начиная обработку с корня: в текущем узле с заданной вероятностью принимается решение об усечении дерева в этом узле (узел становится листом).

Аргументы: *произвольное бинарное дерево*;
вероятность усечения;
результирующее дерево.

```
?- pred(s(f(b(m,k),a),t(a,g)),0.5,X).
X = s(f,t(a,g))
yes
?- pred(s(f(b(m,k),a),t(a,g)),0.5,X).
X = s(f(b,a),t)
yes
?- pred(s(f(b(m,k),a),t(a,g)),0.5,X).
X = s
yes
?-
```

13. Заданный узел в дереве сделать корнем дерева с той же связностью узлов, что и в исходном дереве. Результирующее дерево не является бинарным.

Аргументы: *произвольное бинарное дерево*;
имя узла;
результирующее дерево.

```
?- pred(s(f(b(m,k),w),t(a,g)),s,X).
X = s(f(b(m,k),w),t(a,g))
yes
?- pred(s(f(b(m,k),w),t(a,g)),f,X).
X = f(b(m,k),w,s(t(a,g)))
yes
?- pred(s(f(b(m,k),w),t(a,g)),a,X).
X = a(t(s(f(b(m,k),w)),g)
yes
?-
```

14. Подсчитать количество узлов дерева, лежащих на заданной глубине.

Аргументы: *произвольное бинарное дерево*;
необходимая глубина;
количество узлов.

```
?- pred(s(f(b(m,k),a),t(a,w)),1,X).
X = 1
yes
?- pred(s(f(b(m,k),a),t(a,w)),3,X).
X = 4
yes
?- pred(s(f(b(m,k),a),t(a,w)),4,X).
X = 2
yes
?-
```

15. Произвести обработку дерева в каждом узле по следующему правилу. Если узел имеет заданное имя, то все его дочерние узлы должны стать дочерними узлами его родительского узла, а сам этот узел удаляется из дерева.

Аргументы: *произвольное бинарное дерево*;
имя узла;
результирующее дерево.

```
?- pred(s(f(b(b(u(i,o),v),k),a),t(b,g)),b,X).
X = s(f(u(i,o),v,k,a),t(g))
yes
?-
```

16. Подсчитать количество узлов с заданным именем.

Аргументы: *произвольное бинарное дерево*;
имя узла;
количество узлов.

```
?- pred(s(f(b(b(u(i,o),v),k),a),t(b,g)),b,X).
X = 3
yes
?-
```

17. Определить путь между двумя заданными узлами.

Аргументы: *произвольное бинарное дерево*;
имя первого узла;
имя второго узла;
путь (*список*).

```
?- pred(s(f(w(b(u(i,o),v),k),a),t(r,g)),w,r,X).
X = [w,f,s,t,r]
yes
?- pred(s(f(w(b(u(i,o),v),k),a),t(r,g)),i,o,X).
X = [i,u,o]
yes
?-
```

18. Переименовать все узлы, имеющие заданное имя.

Аргументы: *произвольное бинарное дерево*;
старое имя узла;
новое имя узла;
результирующее дерево.

```
?- pred(s(f(b(b(u(i,o),v),k),a),t(b,g)),b,r,X).
X = s(f(r(r(u(i,o),v),k),a),t(r,g))
yes
?-
```

19. Найти все поддеревья заданного дерева.

Аргументы: *произвольное бинарное дерево*;
поддерево.

```
?- pred(a(f(a(m,k),r),n(i,o)),X).
X = a(f(a(m,k),r),n(i,o)) -> ;
X = f(a(m,k),r) -> ;
X = a(m,k) -> ;
X = m -> ;
X = k -> ;
X = r -> ;
X = n(i,o) -> ;
X = i -> ;
X = o -> ;
no
?-
```

20. Найти все пути в дереве от корня до его листьев.
Аргументы: *произвольное бинарное дерево*;
путь (*список*).

```
?- pred(a(f(a(m,k),r),n(i,o)),X).
X = [a,f,a,m] -> ;
X = [a,f,a,k] -> ;
X = [a,f,r] -> ;
X = [a,n,i] -> ;
X = [a,n,o] -> ;
no
?-
```

21. Найти все пути от корня до его листьев, лежащих на максимальной глубине.
Аргументы: *произвольное бинарное дерево*;
путь (*список*).

```
?- pred(a(f(a(m,k),r),n(i,o)),X).
X = [a,f,a,m] -> ;
X = [a,f,a,k] -> ;
no
?-
```

22. Найти все поддеревья, имеющие глубину не больше заданной.
Аргументы: *произвольное бинарное дерево*;
максимальная глубина;
поддерево.

```
?- pred(a(f(a(m,k),r),n(i,o)),2,X).
X = a(m,k) -> ;
X = m -> ;
X = k -> ;
X = r -> ;
X = n(i,o) -> ;
X = i -> ;
X = o -> ;
no
?-
```

23. Выполнить перебор листьев в порядке убывания их глубины.
Аргументы: *произвольное бинарное дерево*;
лист дерева.

```
?- pred(a(f(a(m,k),r),n(i(d,e),o)),X).
X = m -> ;
X = k -> ;
X = d -> ;
X = e -> ;
X = r -> ;
X = o -> ;
no
?-
```

24. Произвести приведение полинома.
Аргументы: *исходный полином*;
результирующий полином.

```
?- pred(2+6*x^3-7*x^2-4+x^3+6*x^2,X).
X = -2-x^2+7*x^3
yes
?-
```

25. Произвести символьное дифференцирование полинома, который задается структурой вида: *a*+*b***x*+*c***x*^2+*d***x*^3+...
Аргументы: *исходный полином*;
результирующий полином.

```
?- pred(2+6*x-7*x^3,X).
X = 6-21*x^2
yes
?-
```

26. Произвести деление полиномов. Исходные полиномы задаются структурами вида: *a+b*x+c*x^2+d*x^3+...*

Аргументы: *первый полином*;
второй полином;
результирующий полином, остаток от деления.

```
?- pred(2+6*x-7*x^3,1+x,X,Y).
X = -7*x^2+7*x-1
Y = 3
yes
?-
```

27. Расширить синтаксис языка Пролог, введя операции (см. встроенный предикат `op/3`) над множествами (списки). Следует использовать следующие имена операций: «`iss`» – операция извлечения результата; «+» – операция объединения; «`*`» – операция пересечения; «`-`» – операция вычитания.

Аргументы: *арифметическое выражение* (*в арифметике множеств*);
свободная переменная.

```
?- X iss [a,s,d,f] * [d,g,f] + [p,f].
X = [d,f,p]
yes
?-
```

28. Расширить синтаксис языка Пролог, введя операции (см. встроенный предикат `op/3`) модульной арифметики. Следует использовать имя «`ism`» для операции извлечения результата и стандартные «`+, -, *, /`» – для арифметических операций. Для хранения модуля, по которому производятся вычисления, необходимо использовать факт `ism/1`. В качестве модуля предполагается использовать только простые числа.

Аргументы: *арифметическое выражение*;
свободная переменная.

```
?- ism(X).
X = 3
yes
?- X ism (24+38)/2.
X = 1
yes
?-
```

29. Произвести перестановку поддеревьев в каждом узле дерева.

Аргументы: *произвольное бинарное дерево*;
результирующее дерево.

```
?- pred(s(f(b(m,k),a),f(a,g)),X).
X = s(f(g,a),f(a,b(m,k)))
yes
?-
```

Описание шкалы оценивания:

Оценка (баллы)	**Критерии оценки**
Отлично 7	Лабораторная работа выполнена в срок. Собеседование по лабораторной работе показало владение теоретическим материалом и свободное использование терминологии Пролога при объяснении работы разработанной процедуры.
Хорошо 6	Лабораторная работа выполнена с небольшой задержкой (не более недели). Собеседование по лабораторной работе показало владение теоретическим материалом и свободное использование терминологии Пролога при объяснении работы разработанной процедуры.
Удовлетворительно 4	Лабораторная работа выполнена с большой задержкой (более недели). Собеседование по лабораторной работе показало владение теоретическим материалом и свободное использование терминологии Пролога при объяснении работы разработанной процедуры.

Удовлетворительно 5	Собеседование по лабораторной работе показало слабое владение теоретическим материалом и затруднения в использовании терминологии Пролога при объяснении работы разработанной процедуры.
Неудовлетворительно 0–3	Либо лабораторная работа не выполнена, либо обнаружено полное непонимание разработанной процедуры.

4.9. Лабораторная работа № 4

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
Обнинский институт атомной энергетики –
филиал федерального государственного автономного образовательного учреждения высшего профессионального образования «Национальный исследовательский ядерный университет «МИФИ»

Кафедра автоматизированных систем управления

Комплект заданий к лабораторной работе

по дисциплине Логическое программирование

Темы: «Структуры»

1. Определить наличие на каком-либо из путей от корня до листа хотя бы двух узлов с одинаковым именем.

Аргументы: *произвольное дерево*.

```
?- pred(a(f(i(m,k),r,a(t)),n)).
yes
?- pred(a(f(i(m,k),r,o(t)),n)).
no
?-
```

2. Определить наличие хотя бы двух узлов с одинаковым именем на одной глубине.

Аргументы: *произвольное дерево*.

```
?- pred(s(f(b(m,k),a),n(b,g),r(u))).
yes
?- pred(s(f(b(m,k),a),n(t,g),r(u))).
no
?-
```

3. Определить наличие двух одинаковых путей от корня до листа.

Аргументы: *произвольное дерево*.

```
?- pred(s(f(b(m,k),a),f(a,g),n(h))).
yes
?- pred(s(f(y(m,k),a),f(t,g),n(h))).
no
?-
```

4. Определить глубину дерева.

Аргументы: *произвольное дерево*;
глубина дерева.

```
?- pred(s(f(b(m(i),k),a),f(a,g),h(y)),X).
X = 5
yes
?-
```

5. Определить, являются ли два заданных дерева равными с точностью до перестановки поддеревьев в каждом узле.

Аргументы: *первое дерево*;
второе дерево.

```
?- pred(s(t(b(m,k),a),f(a,g)), s(t(b(w,k,m),a),f(g))).
yes
?- pred(s(t(b(m,k),a),f(a,g)), s(t(a,b(k,m,w)),f(g))).
yes
?-
```

6. Собрать все узлы в список.

Аргументы: *произвольное дерево*;
список узлов.

```
?- pred(s(f(b(m,k,n),a(r)),f(a,g)),X).
X = [s,f,b,m,k,n,a,f,a,r,g]
yes
?-
```

7. Найти максимум количества узлов, лежащих на одной глубине.

Аргументы: *произвольное дерево*;
количество узлов.

```
?- pred(s(f(b(m,k(e),n),a),t(r,w)),X).
X = 4
yes
?-
```

8. Обрезать дерево на заданной глубине.

Аргументы: *произвольное дерево*;
необходимая глубина;
результирующее дерево.

```
?- pred(s(f(b(m,k),a),t(a,g),j(u)),2,X).
X = s(f,t,j)
yes
?- pred(s(f(b(m,k),a),t(a,g),j(u)),3,X).
X = s(f(b,a),t(a,g),j(u))
yes
?-
```

9. Собрать в список имена всех узлов дерева, лежащих на заданной глубине.

Аргументы: *произвольное дерево*;
необходимая глубина;
список узлов.

```
?- pred(s(f(b(m,k),a),t(a,g),j(u)),1,X).
X = [s]
yes
?- pred(s(f(b(m,k),a),t(a,g),j(u)),3,X).
X = [b,a,a,g,u]
yes
?- pred(s(f(b(m,k),a),t(a,g),j(u)),5,X).
X = []
yes
?-
```

10. Произвести следующую процедуру над деревом, начиная обработку с корня: в текущем узле с заданной вероятностью принимается решение об усечении дерева в этом узле (узел становится листом).

Аргументы: *произвольное дерево*;
вероятность усечения;
результирующее дерево.

```
?- pred(s(f(b(m,k),a),t(a)),0.5,X).
X = s(f,t(a))
yes
?- pred(s(f(b(m,k),a),t(a)),0.5,X).
X = s(f(b,a),t)
yes
?- pred(s(f(b(m,k),a),t(a)),0.5,X).
X = s
yes
?-
```

11. Найти в дереве все поддеревья, являющиеся бинарными.
Аргументы: *произвольное дерево*;
поддерево.

```
?- pred(a(f(a(m,k(v)),r),n(i(d,e,z(q,w)),o)),X).
X = z(q,w) -> ;
no
?-
```

12. Сделать в дереве заданный узел корнем дерева с той же связностью узлов, что и в исходном дереве.
Аргументы: *произвольное дерево*;
имя узла;
результирующее дерево.

```
?- pred(s(f(b(m,k),w),t(a)),s,X).
X = s(f(b(m,k),w),t(a))
yes
?- pred(s(f(b(m,k),w),t(a)),f,X).
X = f(b(m,k),w,s(t(a)))
yes
?- pred(s(f(b(m,k),w),t(a)),a,X).
X = a(t(s(f(b(m,k),w)))
yes
?-
```

13. Подсчитать количество узлов дерева, лежащих на заданной глубине.
Аргументы: *произвольное дерево*;
необходимая глубина;
количество узлов.

```
?- pred(s(f(b(m,k),a),t(a)),1,X).
X = 1
yes
?- pred(s(f(b(m,k),a),t(a)),3,X).
X = 3
yes
?- pred(s(f(b(m,k),a),t(a)),4,X).
X = 2
yes
?-
```

14. Подсчитать количество узлов с заданным именем.
Аргументы: *произвольное дерево*;
имя узла;
количество узлов.

```
?- pred(s(f(b(b(u(i,o),v),k),a),t(b,g,b(i))),b,X).
X = 4
yes
?-
```

15. Произвести обработку дерева в каждом узле по следующему правилу. Если узел имеет заданное имя, то все его дочерние узлы должны стать дочерними узлами его родительского узла, а сам этот узел удаляется из дерева.
Аргументы: *произвольное дерево*;
имя узла;
результирующее дерево.

```
?- pred(s(f(b(b(u(i,o),v),k),a),t(b,g),b),b,X).
X = s(f(u(i,o),v,k,a),t(g))
yes
?-
```

16. Определить путь между двумя заданными узлами.
Аргументы: *произвольное дерево*;
имя первого узла:
имя второго узла;
путь (*список*).

```
?- pred(s(f(w(b(u(i,o,d),v),k),a),t(r(u),g),m),w,r,X).
X = [w,f,s,t,r]
yes
?- pred(s(f(w(b(u(i,o,d),v),k),a),t(r(u),g),m),i,d,X).
X = [i,u,d]
yes
?-
```

17. Переименовать все узлы, имеющие заданное имя.
Аргументы: *произвольное дерево*;
старое имя узла;
новое имя узла;
результирующее дерево.

```
?- pred(s(f(b(b(u(i,o),v),k),a),t(b(e),g),b),b,r,X).
X = s(f(r(r(u(i,o),v),k),a),t(r(e),g),r)
yes
?-
```

18. Найти все поддеревья.
Аргументы: *произвольное дерево*;
поддерево.

```
?- pred(a(f(a(m,k,v),r),n(i)),X).
X = a(f(a(m,k,v),r),n(i)) -> ;
X = f(a(m,k,v),r) -> ;
X = a(m,k,v) -> ;
X = m -> ;
X = k -> ;
X = v -> ;
X = r -> ;
X = n(i) -> ;
X = i -> ;
no
?-
```

19. Найти в дереве все пути от корня до его листьев.
Аргументы: *произвольное дерево*;
путь (*список*).

```
?- pred(a(f(a(m,k),r),n(i,o,v(x))),X).
X = [a,f,a,m] -> ;
X = [a,f,a,k] -> ;
X = [a,f,r] -> ;
X = [a,n,i] -> ;
X = [a,n,o] -> ;
X = [a,n,v,x] -> ;
no
?-
```

20. Найти в дереве все пути от корня до его листьев, лежащих на максимальной глубине.
Аргументы: *произвольное дерево*;
путь (*список*).

```
?- pred(a(f(a(m,k,v),r),n(i,o)),X).
X = [a,f,a,m] -> ;
X = [a,f,a,k] -> ;
X = [a,f,a,v] -> ;
no
?-
```

21. Найти в дереве все поддеревья, имеющие глубину не больше заданной.
Аргументы: *произвольное дерево*;
максимальная глубина;
поддерево.

```
?- pred(a(f(a(m,k),r,v(g)),n(i,o)),2,X).
X = a(m,k) -> ;
X = m -> ;
X = k -> ;
X = r -> ;
X = v(g) -> ;
X = g -> ;
X = n(i,o) -> ;
X = i -> ;
X = o -> ;
no
?-
```

22. Выполнить перебор листьев в порядке убывания их глубины.
Аргументы: *произвольное дерево*;
лист дерева.

```
?- pred(a(f(a(m,k(v)),r),n(i(d,e,z),o)),X).
X = v -> ;
X = m -> ;
X = k -> ;
X = d -> ;
X = e -> ;
X = z -> ;
X = r -> ;
X = o -> ;
no
?-
```

23. Найти узел, имеющий максимальное количество потомков.
Аргументы: *произвольное дерево*;
узел дерева.

```
?- pred(a(f(a(m,k(v)),r),n(i(d,e,z(q,w,t)),o)),X).
X = i -> ;
X = z -> ;
no
?-
```

24. Составить описание каждого узла дерева (потомки нумеруются слева направо).
Аргументы: *произвольное дерево*;
описание узла (*атом*).

```
?- pred(a(f(a(m,k(v)),r),n(i(d,e,z)),o),X).
X = a -> ;
X = 1:f -> ;
X = 1:1:a -> ;
X = 1:1:1:m -> ;
X = 1:1:2:k -> ;
X = 1:1:2:1:v -> ;
X = 1:2:r -> ;
X = 2:n -> ;
X = 2:1:i -> ;
X = 2:1:1:d -> ;
X = 2:1:2:e -> ;
X = 2:1:3:z -> ;
X = 3:o -> ;
no
?-
```

25. Дана строка, которая с помощью предиката `string_term/2` (см. пример) корректно преобразуется в терм. Предполагается, что строка может преобразовываться в структуру любой арности, содержащую свободные переменные. Собрать из исходной строки в список имена свободных переменных.

Аргументы: *исходная строка*;
список имен переменных.

```
?- string_term('r(T,m(T,K))',B).
B = r(_70,m(_70,_84))
yes
?- pred('r(T,m(T,K))',X).
X = ['T','K']
yes
?-
```

26. Сформировать графическое представление дерева в виде строки, используя символы псевдографики.

Аргументы: *произвольное дерево*;
строка.

```
?- pred(a(f(a(m,k(v)),r),n(i(d,e,z)),o),X).
X =
a
├─f
│ ├─a
│ │ ├─m
│ │ └─k
│ │   └─v
│ └─r
├─n
│ └─i
│   ├─d
│   ├─e
│   └─z
└─o
yes
?-
```

Описание шкалы оценивания:

Оценка (баллы)	**Критерии оценки**
Отлично 11-12	Лабораторная работа выполнена в срок. Собеседование по лабораторной работе показало владение теоретическим материалом и свободное использование терминологии Пролога при объяснении работы разработанной процедуры.
Хорошо 9-10	Лабораторная работа выполнена с небольшой задержкой (не более недели). Собеседование по лабораторной работе показало владение теоретическим материалом и свободное использование терминологии Пролога при объяснении работы разработанной процедуры.
Удовлетворительно 8	Лабораторная работа выполнена с большой задержкой (более недели). Собеседование по лабораторной работе показало владение теоретическим материалом и свободное использование терминологии Пролога при объяснении работы разработанной процедуры.
Удовлетворительно 7	Собеседование по лабораторной работе показало слабое владение теоретическим материалом и затруднения в использовании терминологии Пролога при объяснении работы разработанной процедуры.
Неудовлетворительно 0–6	Либо лабораторная работа не выполнена, либо обнаружено полное непонимание разработанной процедуры.

4.10. Лабораторная работа № 5

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
Обнинский институт атомной энергетики –
филиал федерального государственного автономного образовательного учреждения высшего профессионального образования «Национальный исследовательский ядерный университет «МИФИ»

Кафедра автоматизированных систем управления

Комплект заданий к лабораторной работе

по дисциплине Логическое программирование

Темы: «Интерпретатор и способы управления его работой», «Организация термов в базы данных»

1. Если нумеровать потомков каждого узла дерева слева направо, то можно для каждого узла составить его описание. В разделе БД хранится терм, представленный в виде совокупности описаний узлов. Преобразовать описание в структуру Пролога.

Аргументы: *имя раздела БД* (*имя структуры*);
структура.

```
?- recorded(tst,X,_).
X = 1:f -> ;
X = 1:1:a -> ;
X = 1:1:1:m -> ;
X = 1:1:2:k -> ;
X = 1:1:2:1:v -> ;
X = 1:2:r -> ;
X = 2:n -> ;
X = 2:1:i -> ;
X = 2:1:1:d -> ;
X = 2:1:2:e -> ;
X = 2:1:3:z -> ;
X = 3:o -> ;
no
?- pred(tst,X).
X = tst(f(a(m,k(v)),r),n(i(d,e,z)),o)
yes
?-
```

2. Растровое изображение хранится в дисковом файле формата PPM. Построить зеркальное отражение исходного изображения. Результат сохранить в новом файле.

Аргументы: *имя исходного файла*;
имя нового файла.

```
?- pred('sample.ppm','new.ppm').
yes
?-
```

3. В разделе БД хранится терм в графическом представлении. Преобразовать его в структуру Пролога.

Аргументы: имя раздела БД;
структура.

```
?- recorded(test,X,_).
X = a -> ;
X = ├─f -> ;
X = │ ├─a -> ;
X = │ │ ├─m -> ;
X = │ │ └─k -> ;
X = │ │   └─v -> ;
X = │ └─r -> ;
X = ├─n -> ;
X = │ └─i -> ;
X = │   ├─d -> ;
X = │   ├─e -> ;
X = │   └─z -> ;
X = └─o -> ;
no
?- pred(test,X).
X = a(f(a(m,k(v)),r),n(i(d,e,z)),o)
yes
?-
```

4. Растровое изображение, хранится в дисковом файле формата PPM. Преобразовать исходное цветное изображение в черно-белое (в градациях серого). Результат сохранить в новом файле. Для пересчета цветовых координат пикселя (*R*, *G*, *B*) к яркости можно воспользоваться формулой $L = 0.3R + 0.59G + 0.11B$.

Аргументы: *имя исходного файла*;
имя нового файла.

```
?- pred('sample.ppm','new.ppm').
yes
?-
```

5. В разделе БД хранится текст на русском языке. Каждый терм, хранящийся в разделе, представляет собой предложение на русском языке. Осуществить форматирование текста, переписав его в новый раздел БД. Форматирование производится путем разбиения всего текста на строки длиной, не превышающей заданную ширину страницы (переносы расставляются по правилам русского языка).

Аргументы: *имя раздела БД с исходным текстом*;
ширина страницы (*в знаках*);
имя раздела БД с отформатированным текстом.

```
?- recorded(test1,X,_).
X = Впрочем, был еще и повод. -> ;
X = Возможно, главный. -> ;
X = Его сложно объяснить в двух словах, мимоходом. -> ;
X = И чем ближе к концу моего рассказа, тем сложнее. -> ;
no
?- pred(test1,46,test2).
yes
?- recorded(test2,X,_).
X = Впрочем, был еще и повод. Возможно, главный. -> ;
X = Его сложно объяснить в двух словах, мимоходом. -> ;
X = И чем ближе к концу моего рассказа, тем слож- -> ;
X = нее. -> ;
no
?-
```

6. В раздел БД загрузить текст программы на языке Паскаль и удалить из него комментарии.

Аргументы: *имя файла с исходной программой*;
имя раздела БД.

```
?- pred('first.pas',test1).
yes
?-
```

7. В двух разделах БД хранятся произвольные термы. Переписать в третий раздел только те термы, которые содержатся в первом разделе, но не содержатся во втором.

Аргументы: *имя первого раздела БД*;
имя второго раздела БД;
имя третьего раздела БД.

```
?- recorded(test1,X,_).
X = a -> ;
X = v -> ;
X = t -> ;
X = 7 -> ;
no
?- recorded(test2,X,_).
X = 4 -> ;
X = 6 -> ;
X = a -> ;
X = x -> ;
no
?- pred(test1,test2, test3).
yes
?- recorded(test3,X,_).
X = v -> ;
X = t -> ;
X = 7 -> ;
no
?-
```

8. Растровое изображение хранится в дисковом файле формата PPM. Подсчитать количество цветов, используемых в данном изображении.

Аргументы: *имя файла*;
количество цветов.

```
?- pred('sample.ppm',X).
X = 27 -> ;
yes
?-
```

9. В разделе БД содержатся существительные русского языка. Заполнить заданный шаблон этими существительными так, что бы каждое слово было использовано всего один раз.

Аргументы: *имя раздела БД* (*словарь*);
имя факта (*задает шаблон*);
список слов (*заполняют шаблон*).

```
?- pred(verb,form,X).
X = казак*роман*бокал+короб*замок*канал -> ;
…
yes
?-
```

10. В разделе БД хранится шаблон кроссворда. Преобразовать шаблон к виду, пригодному для использования в задаче составления кроссворда.

Аргументы: *имя раздела БД*;
шаблон (*список*).

```
?- recorded(test1,X,_).
X = xxxxx -> ;
X = x x x -> ;
X = xxxxx -> ;
X = x x x -> ;
X = xxxxx -> ;
no
?- pred(test1,X).
X = [_1,_2,_3,_4,_5]*[_6,_7,_8,_9,_A]*[_B,_C,_D,_E,_F]+
[_1,_11,_9,_12,_5]*[_3,_13,_8,_14,_D]*[_5,_15,_A,_16,_F]
yes
?-
```

11. В разделе БД содержатся числа. Найти все различные совокупности чисел (как по количеству элементов, так и по составу) такие, что сумма чисел совокупности не превышает заданного числа.

Аргументы: *имя раздела БД* (*множество чисел*);
ограничение на сумму (*число*);
список ссылочных номеров допустимого подмножества.

```
?- recorded(test1,X,R).
X = 3
R = ~402F0A -> ;
X = 2
R = ~402F32 -> ;
X = 5
R = ~402F5E -> ;
X = 7
R = ~402F20 -> ;
X = 2
R = ~402F14 -> ;
no
?- pred(test1,5,X).
X =  [~402F0A]-> ;
X =  [~402F32]-> ;
X =  [~402F5E]-> ;
X =  [~402F14]-> ;
X =  [~402F0A,~402F32]-> ;
X =  [~402F0A,~402F14]-> ;
X =  [~402F32,~402F14]-> ;
no
?-
```

12. Растровое изображение хранится в дисковом файле формата PPM. Произвести замену цвета, указанного во втором аргументе, на цвет, указанный в третьем аргументе. Результат сохранить в новом файле.

Аргументы: *имя исходного файла*;
заменяемый цвет; *новый цвет*
имя нового файла.

```
?- pred('sample.ppm',(32,120,200),(0,0,0),'new.ppm').
yes
?-
```

13. Дана матрица инцидентности графа. Сформировать раздел БД так, чтобы каждой вершине графа соответствовал свой терм БД, являющийся списком ссылочных номеров термов, соответствующих тем вершинам графа, которые инцидентны данной вершине графа. Для примера диалога в среде интерпретатора использованы граф и матрица инцидентности из приложения В.

Аргументы: *матрица инцидентности*;
имя раздела БД.

```
?- pred([[1,0,0,1],[1,1,0,0],[0,1,1,0],[0,0,1,1]],m).
yes
?- recorded(m,X,R).
X = [~402F0A,~402F32]
R = ~402F14 -> ;
X = [~402F14,~402F5E]
R = ~402F0A -> ;
X = [~402F0A,~402F32]
R = ~402F5E -> ;
X = [~402F14,~402F5E]
R = ~402F32 -> ;
no
?-
```

Описание шкалы оценивания:

Оценка (баллы)	**Критерии оценки**
Отлично 11-12	Лабораторная работа выполнена в срок. Собеседование по лабораторной работе показало владение теоретическим материалом и свободное использование терминологии Пролога при объяснении работы разработанной процедуры.
Хорошо 9-10	Лабораторная работа выполнена с небольшой задержкой (не более недели). Собеседование по лабораторной работе показало владение теоретическим материалом и свободное использование терминологии Пролога при объяснении работы разработанной процедуры.
Удовлетворительно 8	Лабораторная работа выполнена с большой задержкой (более недели). Собеседование по лабораторной работе показало владение теоретическим материалом и свободное использование терминологии Пролога при объяснении работы разработанной процедуры.
Удовлетворительно 7	Собеседование по лабораторной работе показало слабое владение теоретическим материалом и затруднения в использовании терминологии Пролога при объяснении работы разработанной процедуры.
Неудовлетворительно 0–6	Либо лабораторная работа не выполнена, либо обнаружено полное непонимание разработанной процедуры.