34.2

implement main  
    open core, console  
  
class predicates  
    ack:(unsigned, unsigned, unsigned [out]) determ.  
  
clauses  
    ack(0, N, Result) :-  
        Result = N + 1, !.  
  
    ack(M, 0, Result) :-  
        M > 0, !,  
        ack(M-1, 1, Result).  
  
    ack(M, N, Result) :-  
        M > 0, N > 0, !,  
        ack(M, N-1, Temp),  
        ack(M-1, Temp, Result).  
  
    run() :-  
        if ack(3, 4, R) then  
            write("ack(3,4) = ", R), nl  
        else  
            write("Ошибка вычисления"), nl  
        end if,  
        \_ = readline().  
  
end implement main  
  
goal  
    console::run(main::run).

34.10

implement main  
    open core, console, string, math  
  
domains  
    знак = минус; плюс.  
  
class predicates  
    sin\_series:(  
        unsigned,    % счетчик  
        real,        % x  
        real,        % аккумулятор  
        real,        % последний член  
        знак,        % знак  
        unsigned,    % число членов  
        real [out]) determ.  % результат  
  
clauses  
    sin\_series(\_,\_,Аккум,\_,\_,0,Аккум) :- !.  
  
    sin\_series(Счетчик,X,Аккум,Член,минус,N,Результат) :-  
        Член1 = Член \* X \* X / (Счетчик \* (Счетчик+1)),  
        sin\_series(Счетчик+2,X,Аккум-Член1,Член1,плюс,N-1,Результат), !.  
  
    sin\_series(Счетчик,X,Аккум,Член,плюс,N,Результат) :-  
        Член1 = Член \* X \* X / (Счетчик \* (Счетчик+1)),  
        sin\_series(Счетчик+2,X,Аккум+Член1,Член1,минус,N-1,Результат).  
  
    run() :-  
        X = pi / 6,  
        if sin\_series(2, X, X, X, минус, 10, Сумма) then  
            Точное = sin(X),  
            writef(  
                @"x = %  
Сумма ряда (10 членов) = %  
sin(x) точное = %  
Разность = %",  
                X, Сумма, Точное, abs(Сумма - Точное)), nl  
        end if,  
        \_ = readline().  
  
end implement main  
  
goal  
    console::run(main::run).

34.11

implement main  
    open core, console, string, math  
  
domains  
    перенос = unsigned.  
  
class predicates  
    сложение:(string, string, string [out]) determ.  
    сложить:(string, string, перенос, charCount, string, string [out]) determ.  
    степень\_10:(charCount, unsigned [out]) determ.  
  
clauses  
    % Вычисление 10^n  
    степень\_10(0, 1) :- !.  
    степень\_10(Н, Результат) :-  
        Н > 0,  
        степень\_10(Н - 1, Результат\_prev),  
        Результат = Результат\_prev \* 10.  
  
    сложение(Аргумент1, Аргумент2, Сумма) :-  
        Длина1 = string::length(Аргумент1),  
        Длина2 = string::length(Аргумент2),  
        Длина = max(Длина1, Длина2),  
  
        if Длина1 > Длина2 then  
            Аргумент22 = adjustRight(Аргумент2, Длина, "0"),  
            сложить(Аргумент1, Аргумент22, 0, Длина, "", Сумма)  
        elseif Длина1 < Длина2 then  
            Аргумент11 = adjustRight(Аргумент1, Длина, "0"),  
            сложить(Аргумент11, Аргумент2, 0, Длина, "", Сумма)  
        else  
            сложить(Аргумент1, Аргумент2, 0, Длина, "", Сумма)  
        end if.  
  
    % Базовые случаи  
    сложить(\_, \_, 0, 0, Сумма, Сумма) :- !.  
  
    сложить(\_, \_, Перенос, 0, Аккум, Сумма) :-  
        Перенос > 0, !,  
        Сумма = concat(toString(Перенос), Аккум).  
  
    % Рекурсивное сложение блоками (по 9 цифр)  
    сложить(Число1, Число2, Перенос, Длина, Аккум, Сумма) :-  
        % Определяем размер блока (максимум 9 цифр)  
        Размер\_блока = min(9, Длина),  
        Остаток\_длины = Длина - Размер\_блока,  
  
        % Извлекаем блоки справа  
        string::front(Число1, Остаток\_длины, Ост1, Блок1\_стр),  
        string::front(Число2, Остаток\_длины, Ост2, Блок2\_стр),  
  
        % Преобразуем блоки в числа и складываем  
        Блок1 = toTerm(Блок1\_стр),  
        Блок2 = toTerm(Блок2\_стр),  
        Сумма\_блока = Блок1 + Блок2 + Перенос,  
  
        % Вычисляем результат блока и новый перенос  
        степень\_10(Размер\_блока, Макс),  
        Результат\_блока = Сумма\_блока mod Макс,  
        Новый\_перенос = Сумма\_блока div Макс,  
  
        % Форматируем блок с ведущими нулями  
        Формат = format("%%0%dd", Размер\_блока),  
        Блок\_результат = format(Формат, Результат\_блока),  
  
        Новый\_аккум = concat(Блок\_результат, Аккум),  
  
        % Рекурсивно обрабатываем оставшиеся блоки  
        сложить(Ост1, Ост2, Новый\_перенос, Остаток\_длины, Новый\_аккум, Сумма).  
  
    run() :-  
        Аргумент1 = "2343124386865890390432032043094385677",  
        Аргумент2 = "890675897560760965079450943568734876",  
  
        write("Число 1: ", Аргумент1), nl,  
        write("Число 2: ", Аргумент2), nl,  
  
        if сложение(Аргумент1, Аргумент2, Сумма) then  
            write("Сумма:   ", Сумма), nl  
        else  
            write("Ошибка вычисления"), nl  
        end if,  
  
        \_ = readline().  
  
end implement main  
  
goal  
    console::run(main::run).

34.17

implement main  
    open core, console, string  
  
class predicates  
    перевод\_10\_2:(unsigned, string [out]) determ.  
    перевести\_10\_2:(unsigned, string, string [out]) determ.  
  
clauses  
    % Основная функция  
    перевод\_10\_2(Число, Биты) :-  
        перевести\_10\_2(Число, "", Биты).  
  
    % Базовый случай: число стало 0  
    перевести\_10\_2(0, Аккум, Аккум) :- !.  
  
    % Рекурсивное деление на 2  
    перевести\_10\_2(Число, Аккум, Биты) :-  
        Число > 0,  
        Остаток = Число mod 2,  
        Частное = Число div 2,  
        Бит = toString(Остаток),  
        Новый\_аккум = concat(Бит, Аккум),  % Добавляем в начало  
        перевести\_10\_2(Частное, Новый\_аккум, Биты).  
  
    run() :-  
        Число = 10,  
        if перевод\_10\_2(Число, Биты) then  
            write("Десятичное: ", Число), nl,  
            write("Двоичное:   ", Биты), nl  
        else  
            write("Ошибка преобразования"), nl  
        end if,  
        \_ = readline().  
  
end implement main  
  
goal  
    console::run(main::run).