1. Ulchich

1.1. Опишіть вашу предметну область словами

Предметна область «Магазин морозива».

Об’єкти:

1. Морозиво (ідентифікатор, тип, назва, вага, ціна).
2. Магазин (назва, список-асортимент).

Запити:

1. Знайти морозиво в магазині, яке важить більше за задану вагу.
2. Підрахувати, скільки у магазині певного морозива (за ідентифікатором).
3. Повністю викупити певне морозиво (за ідентифікатором).

1.2. Запропонуйте подання [частини] даних Вашої задачі з використанням списків Прологу

Представимо асортимент магазину у вигляді списку. Для цього спочатку опишемо морозиво, не використовуючи списки:

```
predicates
    icecream(icecrm_id, icecrm_type, icecrm_name, icecrm_weight,
             icecrm_price)

domains
    icecrm_id, store_id, icecrm_type = symbol
    icecrm_name = string
    icecrm_weight, icecrm_price = real

clauses
    % морозиво(ID морозива, ім'я, вага, ціна).
    % ID морозива = maxm090 — (Max)i(m)use, (90) g
    % sandwich — брикет, cone — ріжок, bar — ескімо
    icecream(maxm090, sandwich, "Maximuse", 90.0, 18.0).
    icecream(choc100, cone, "Three Chocolates", 100.0, 35.0).
    icecream(mona080, bar, "Monaco Cookies", 80.0, 25.0).
    icecream(sush070, cone, "Super Chocolate", 70.0, 25.0).
```

Тепер представимо асортимент магазину у вигляді списків:

```
domains
    % ... Типи з попереднього завдання
    stock = symbol*

predicates
    % ... Предикати з попереднього завдання
    store(store_id, stock)

clauses
    % ... Факти і правила з попереднього завдання
    % магазин(ID магазину, [асортимент магазину за ID]).
    store(s0, [maxm090, choc100, mona080, sush070]).
    store(s1, [maxm090, mona080, sush070, mona080, maxm090, mona080]).
    store(s2, [choc100, sush070]).
```

1.3. Запропонуйте предикати для розв'язання одного з запитів Вашої задачі з використанням списків Прологу.

Опишемо предикати, що дозволять знайти скільки в магазині певного морозива:

```
predicates
    % ... Предикати з попередніх завдань
    icecrm_count(stock, icecrm_id, integer)
    store_icecrm_count(store_id, icecrm_id, integer)

clauses
    % ... Факти і правила з попереднього завдання
    % Шукає, скільки (`Count`) в списку наявності `Stock`
    % морозива `IceCrmID`.
    % 1. Пустий список: в пустому списку 0 шт. будь-якого морозива
    icecrm_count([], _, 0).
    % 2. Голова співпадає з бажаним морозивом `IceCrmID`:
    % поглиблюємось у рекурсію і збільшуємо лічильник
    icecrm_count([IceCrmID | Tail], IceCrmID, Count) :-
        icecrm_count(Tail, IceCrmID, Tmp),
        Count = Tmp + 1.
```

```
    % 3. Голова не співпадає з бажаним морозивом: не змінюємо
    % лічильник, поглиблюємось у рекурсію
    icecrm_count([Head | Tail], IceCrmID, Count) :-
        Head <> IceCrmID,
        icecrm_count(Tail, IceCrmID, Count).

    % Шукає, скільки (`Count`) морозива `IceCrmID` в магазині `Store`
    store_icecrm_count(StoreID, IceCrmID, Count) :-
        store(StoreID, Stock),
        icecrm_count(Stock, IceCrmID, Count).
```

1.4. Запропонуйте подання [частини] даних Вашої задачі з використанням динамічних баз даних Прологу.

Представимо магазини у вигляді динамічної бази даних. Для цього оголосимо її у відповідному розділі:

```
% Створюємо динамічну базу даних `stores` —
% список існуючих магазинів, де визначений предикат `store`
database - stores
    store(store_id, stock)
```

Тепер оголошено динамічний предикат `store/2`. Факти, оголошені з його допомогою, зберігаються в динамічний базі даних `stores`.

1.5. Запропонуйте предикат(и) для розв'язання одного з запитів Вашої задачі з використанням динамічних баз даних Прологу

Оголосимо предикат, який симулює викуп усього певного морозива в магазині:

```
predicates
    % ... Предикати з попередніх завдань
    % Предикат `store/2` відсутній у цій секції,
    % оскільки він описаний у секції динамічних баз даних
    del_icecrm_all(icecrm_id, stock, stock)
    buy_store_icecrm_all(store_id, icecrm_id)
clauses
    % ... Факти і правила з попередніх завдань
```

```
% Рекурсивний предикат, щоб купити усе морозиво типу IceCrmID
% з асортименту (видалити перший елемент `IceCrmID` зі
% списку)
% 1. Купити будь-яке морозиво з пустого асортименту — пустий асортимент
del_icecrm_all(_, [], []).
% 2. Голова списку — морозиво, яке необхідно видалити: видалити
% поточний елемент `IceCrmID` і його наступні інстанції зі списку
del_icecrm_all(IceCrmID, [IceCrmID | Tail], Res) :-
    del_icecrm_all(IceCrmID, Tail, Res).
% 3. Незалежно від елемента `IceCreamID`, зберігати голови списків, щоб
% не втрачати елементи, що йдуть перед видаленим. Без цього правила
% предикат повертає елементи, що йдуть після останнього видаленого
del_icecrm_all(IceCrmID, [Head | Tail], [Head | Res]) :-
    % Перевірка, що морозиво `IceCrmID` не співпадає з головою списку,
    % тобто випадок не підходить під правило 2
    IceCrmID <> Head,
    del_icecrm_all(IceCrmID, Tail, Res).

% Придбати морозиво `IceCrmID` в магазині `StoreID`
% Якщо морозиво придбали, воно видаляється з асортименту
buy_store_icecrm_all(StoreID, IceCrmID) :-
    store(StoreID, Stock),
    del_icecrm_all(IceCrmID, Stock, NewStock),
    retract(
           store(StoreID, _)
    ),
    % Додати магазин StoreID з асортиментом Stock
    % в кінець динамічної БД `stores`
    assertz(
           store(StoreID, NewStock)
    ).
```

1.6. Запропонуйте опис [частини] даних Вашої задачі з використанням засобів мови Ліспо.

Представимо відомості про магазин мовою Ліспо. Мовою Пролог вони описані так:

```
store(s0, [maxm090, choc100, mona080, sush070]).
store(s1, [maxm090, mona080, sush070, mona080, maxm090, mona080]).
store(s2, [choc100, sush070]).
```

Тоді мовою Lisp за допомогою виразу (expression) `let` створимо ідентичні змінні `s0`, `s1`, `s2` і присвоїмо їм відповідний зміст:

```
(let ((s0 '(maxm090 choc100 mona080 sush070))
      (s1 '(maxm090 mona080 sush070 mona080 maxm090 mona080))
      (s2 '(choc100 sush070)))
)
```