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, дешевше за задану ціну.
2. Знайти морозиво в магазині, дорожче за задану ціну.
3. Знайти морозиво в магазині, яке важить більше за задану вагу.
4. Купити морозиво.

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 = maxm100 — (Max)i(m)use, (100) 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]).
    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 в магазині морозиво, важче заданої маси:

```
predicates
    % ... Предикати з попередніх завдань
    icecrm_in_stock(icecrm_id, stock)
    store_has_icecrm(store_id, icecrm_id)
    store_icecrm_weight_over(store_id, icecrm_id, icecrm_weight)

clauses
    % ... Факти і правила з попереднього завдання
    % Рекурсивна перевірка, чи є морозиво в наявності
    % icecrm_in_stock(ID морозива, наявність)
    icecrm_in_stock(IceCrmID, [IceCrmID | _]).
    icecrm_in_stock(IceCrmID, [_ | T]) :-
        icecrm_in_stock(IceCrmID, T).
```

```
    % Перевірка, чи є морозиво IceCrmID у магазині StoreID
    store_has_icecrm(StoreID, IceCrmID) :-
        store(StoreID, Stock),
        icecrm_in_stock(IceCrmID, Stock).

    % Пошук морозива у магазині, яке важить більше за Weight
    % Запит, щоб знайти морозиво в магазині s0, важчі за 80:
    % store_icecrm_weight_over(s0, IceCrmID, 80).
    store_icecrm_weight_over(StoreID, IceCrmID, Weight) :-
        % Знайти асортимент магазину Stock
        store(StoreID, Stock),
        % Знайти морозиво IceCrmID в асортименті
        icecrm_in_stock(IceCrmID, Stock),
        % Знайти ціну морозива
        icecream(IceCrmID, _, _, IceCrmWeight),
        % Істина, якщо вага більше заданої
        IceCrmWeight > Weight.
```

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

Представимо магазини у вигляді динамічної бази даних:

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

Тепер оголошено динамічний предикат `store/2`. Він використовується так само, як і не-динамічні, але факти, оголошені з його допомогою, зберігаються в динамічній 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 покупку морозива в магазині:

```
predicates
    % ... Предикати з попередніх завдань
```

```
    % Предикат `store/2` відсутній у цій секції,
    % оскільки він описаний у секції динамічних баз даних
    buy_icecrm_one(icecrm_id, stock, stock)
    buy_store_icecrm(store_id, icecrm_id)

clauses
    % ... Факти і правила з попередніх завдань

    % Рекурсивний предикат, щоб купити одне морозиво Item
    % з асортименту (видалити перший елемент `Item` зі
    % списку)
    buy_icecrm_one(_, [], []).
    % Відсікання, щоб отримувати лише один результат
    buy_icecrm_one(Item, [Item | Tail], Tail) :- !.
    buy_icecrm_one(Item, [Head | Tail], [Head | Res]) :-
        buy_icecrm_one(Item, Tail, Res).

    % Придбати морозиво `IceCrmID` в магазині `StoreID`
    % Якщо морозиво придбали, воно видаляється з асортименту
    buy_store_icecrm(StoreID, IceCrmID) :-
        store(StoreID, Stock),
        buy_icecrm_one(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]).
```

```
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))
      (s2 '(choc100 sush070)))
)
```