Мультипарадигменне програмування

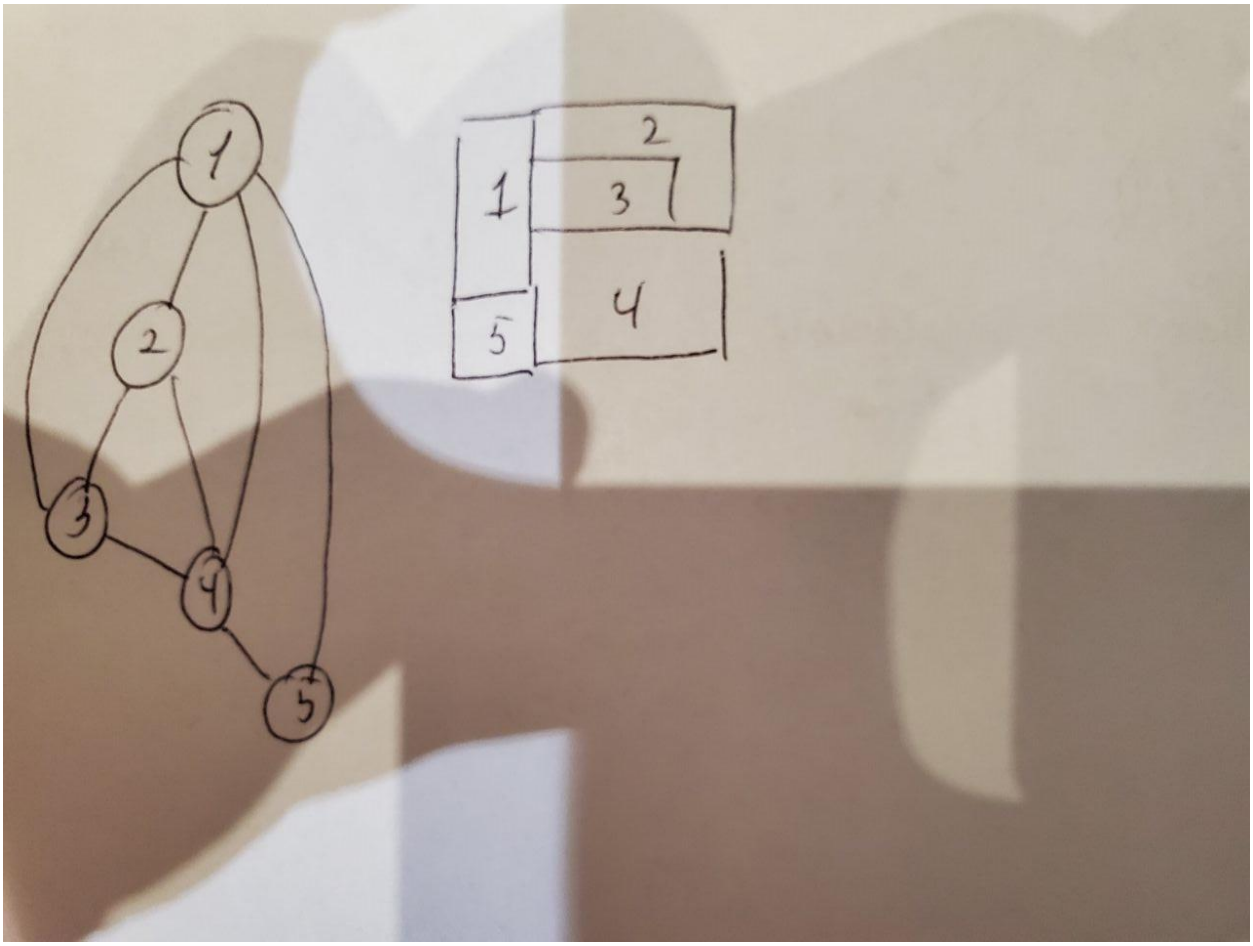
Виконав: Лазанчук К.В.

Перевірив: Очеретяний Олександр Костянтинович

Завдання 2.1

Карту можна розглядати як граф з’єднань, а «вузли» та «ребра» є загальними термінами для цих графів. Вузли, що означають області карти, та будучи суміжними на зображенні, створюють такі з’єднання країв.

Результат на рис. 1.1



Завдання 2.2.1

Для виконання даної частини, нам потрібно взяти код, який був даний у прикладі.

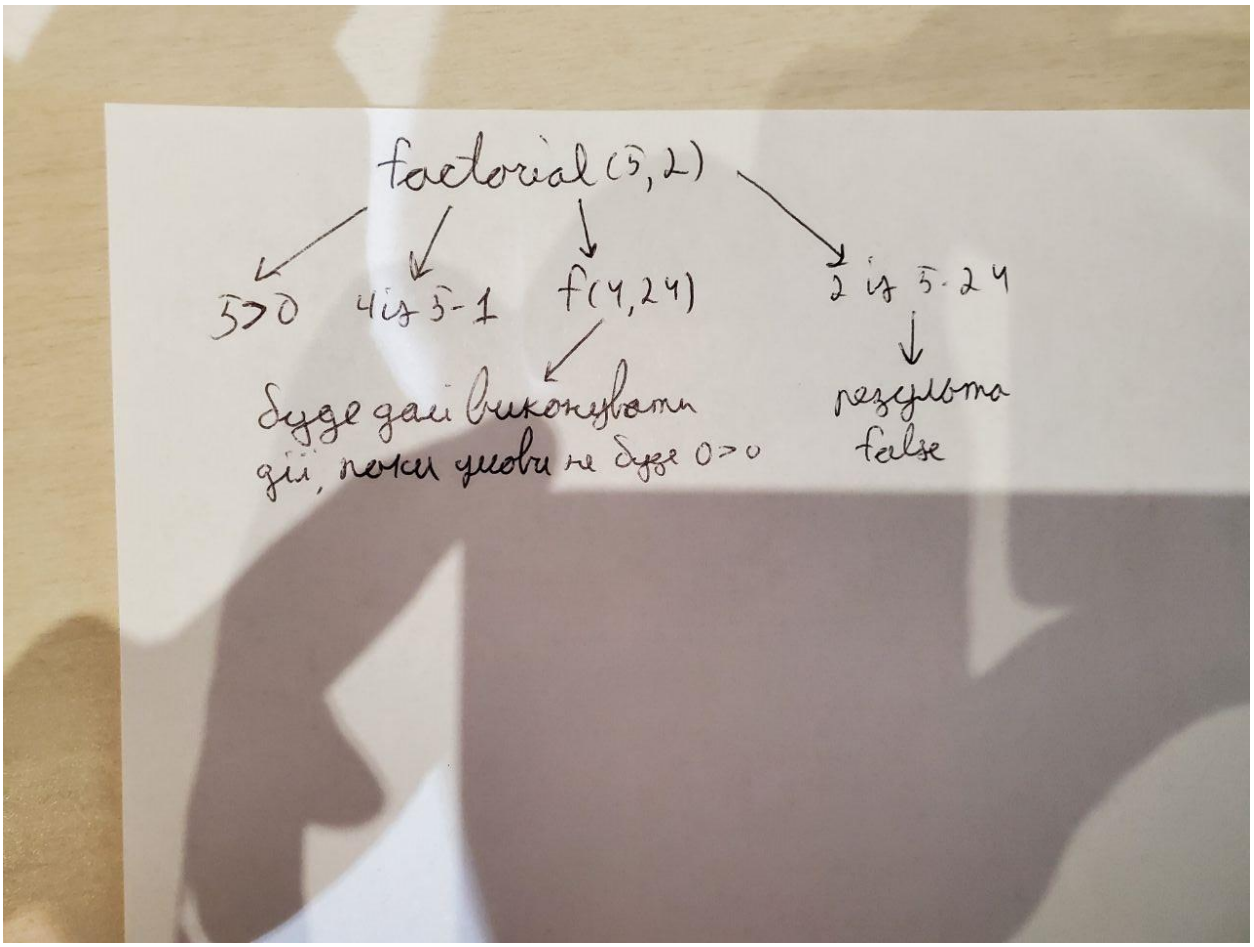
```
factorial(0,1).

factorial(N,F) :-
   N>0,
   N1 is N-1,
   factorial(N1,F1),
   F is N * F1.
```

В умові задачі було сказано, щоб ми підставили значення (5,2).

При підставленні цих даних, програми виведе нам результат «False»

Щоб зрозуміти, чому так сталося, нам потрібно намалювати граф та розібратися. Результат графу на рис. 2.1



Завдання 2.2.2 Для виконання даного завдання, нам потрібно взяти код знову із прикладу.

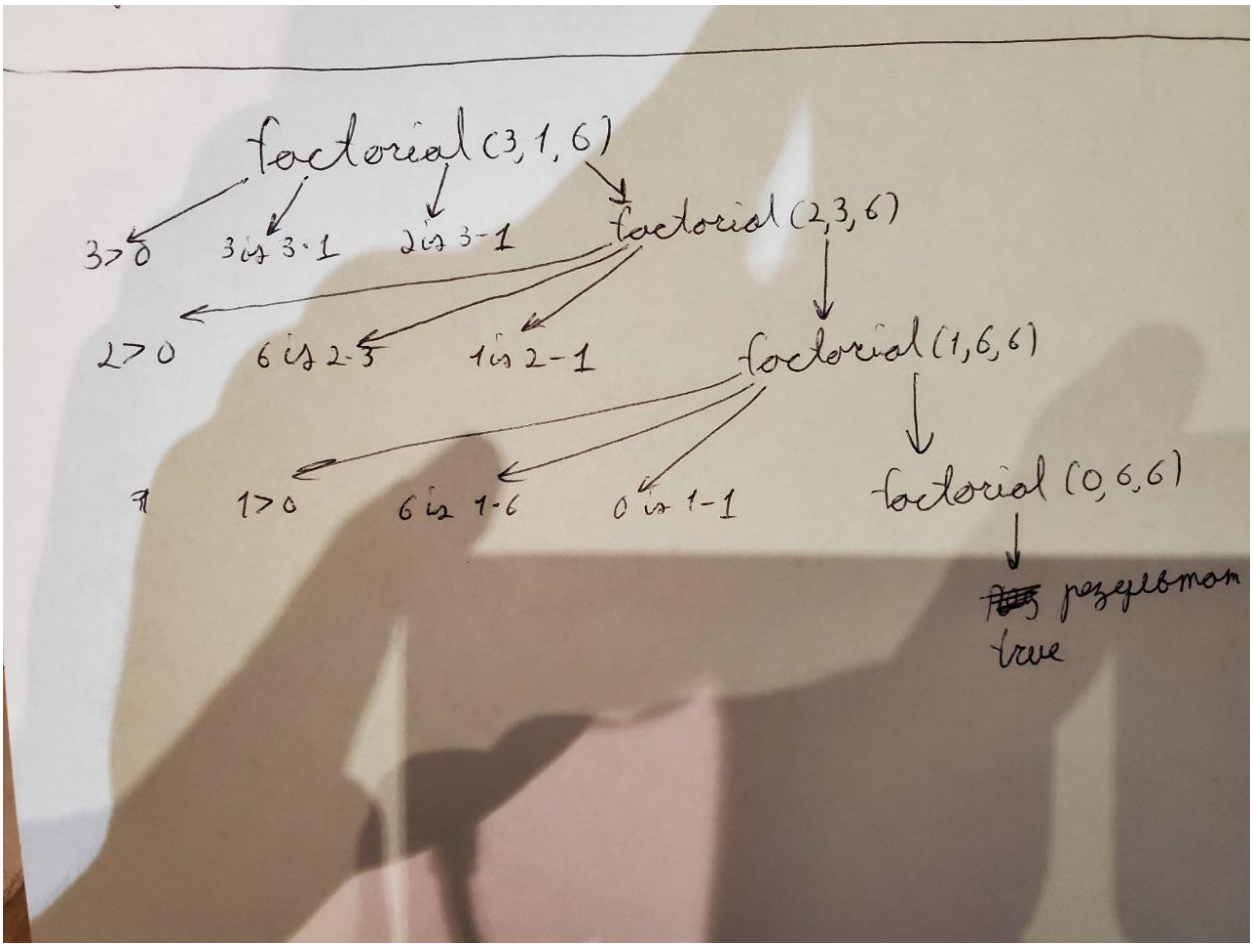
```
factorial(0,F,F).

factorial(N,A,F) :-
    N > 0,
    A1 is N*A,
    N1 is N -1,
    factorial(N1,A1,F).
```

В умові задачі було сказано, щоб ми підставили значення (5,2).

При підставленні цих даних, програми виведе нам результат «true»

Для того, щоб зрозуміти, як вона працюємо, побудуємо графи. Результат графу на рис. 2.2



Результат трасування 'factorial(3,1,6)'

```
Call: factorial(3,1,6)
 Call: 3>0
 Exit: 3>0
 Call: _630 is 3*1
 Exit: 3 is 3*1
 Call: _644 is 3 + -1
 Exit: 2 is 3 + -1
 Call: factorial(2,3,6)
  Call: 2>0
  Exit: 2>0
  Call: _646 is 2*3
  Exit: 6 is 2*3
  Call: _660 is 2 + -1
  Exit: 1 is 2 + -1
  Call: factorial(1,6,6)
   Call: 1>0
   Exit: 1>0
   Call: _662 is 1*6
   Exit: 6 is 1*6
   Call: _676 is 1 + -1
   Exit: 0 is 1 + -1
   Call: factorial(0,6,6)
   Exit: factorial(0,6,6)
  Exit: factorial(1,6,6)
 Exit: factorial(2,3,6)
Exit: factorial(3,1,6)
true
```

Якщо порівнювати програму з завдання 2.2.1 та 2.2.2 то ми можемо побачити, що у першому варіанті, ми спершу рекурсивна викликаємо функцію, а потім коли умова «N>0» вже не проходить то тоді перевіряємо результат «F is N*F1».

У другому варіанті ми спочатку множимо, а потів вже викликаємо функцію для того, щоб зробити ще раз виклик функції до тих пір, поки не дійде до базового результату.

Завдання 2.3.1 Нам потрібно виконати ціль через функцію move(3,left,right,center).

Для виконання даного завдання нам потрібно взяти код знову з прикладу.

```
move(1,X,Y,_) :-
    write('Move top disk from '),
    write(X),
    write(' to '),
    write(Y),
    nl.
move(N,X,Y,Z) :-
    N>1,
    M is N-1,
    move(M,X,Z,Y),
    move(1,X,Y,_),
    move(M,Z,Y,X).
```

Для того, щоб більш добре зрозуміти, як працює данні програма, нам потрібно намалювати граф. Результат на рис. 2.3