CSI 3120

Lab 7 Report

Group #6

Carl Li 300235679
Ruoyu Liu 300176134
Roy Rui 300176548

Task A: Generating Patterns with User Input

Question 1
right_angle_triangle_console/0

- **Purpose:** This predicate starts the process of generating the triangle.
 Prompts the user to enter the height of the triangle.
 Reads the height input and calls print_triangle/2 with the starting row (Current = 1) and the specified height (Height).

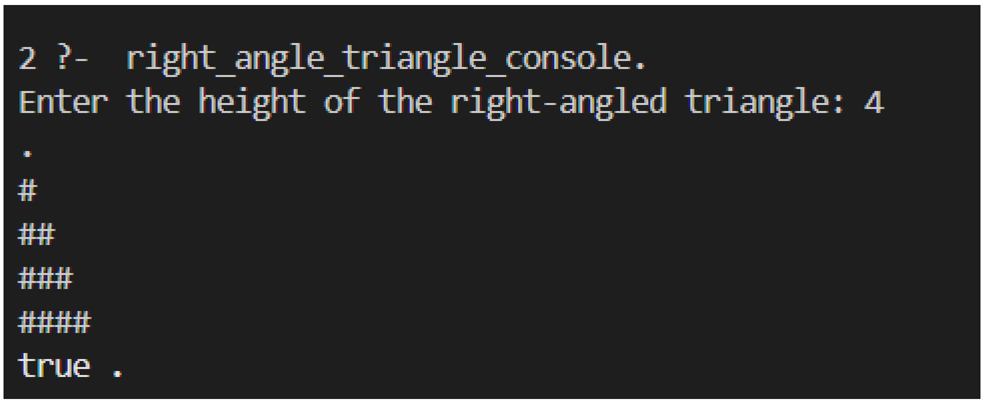
print_triangle/2

- **Purpose:** This predicate recursively prints each row of the triangle.
    - **Base Case:** Stops recursion when the current row number exceeds the height.
    - **Recursive Case:** Calls print_row/1 to print Current number of # symbols for the current row, then increments Current and calls itself for the next row.
- <u>The predicate checks if Current =< Height and continues printing rows. If Current > Height, the recursion ends.</u>

print_row/1

- **Purpose**: This predicate prints N number of # symbols for a single row.
    - **Base Case**: Stops printing when N = 0.
    - **Recursive Case**: Prints one # symbol, decrements N, and calls itself to print the remaining symbols in the row.

Testing:



```
2 ?-  right_angle_triangle_console.
Enter the height of the right-angled triangle: 4
.
#
##
###
####
true .
```

Question 2

isosceles_triangle_pattern_file/2

- **Purpose:** Generates an isosceles triangle pattern of * symbols and writes it to a file.
    - Opens the specified file for writing.
    - Calls write_triangle/3 to generate and write each row to the file.
    - Closes the file and prints a success message using format/2.

write_triangle/3

- **Purpose: Recursively writes rows of the isosceles triangle to the file.**
    - **Base Case: Stops recursion when Current exceeds Height.**
    - **Recursive Case:**
        - Calculates the number of spaces (Spaces) to center-align the triangle.
        - Calculates the number of * symbols (Stars) for the current row.
        - Calls print_spaces/2 and print_stars/2 to write the spaces and stars for the row.
        - Moves to the next row (Next) and calls itself recursively.

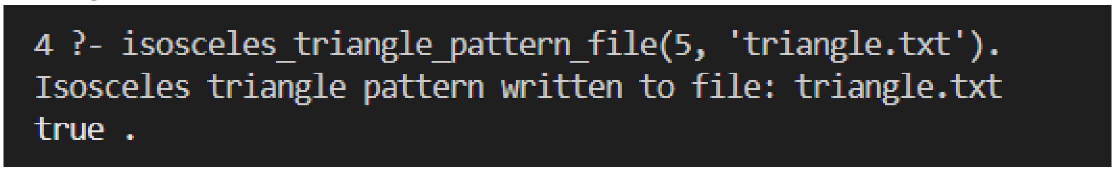
print_spaces/2

- **Purpose**: Writes a specified number of spaces to the file stream.
    - **Base Case**: Stops when there are no spaces to write.
    - **Recursive Case**: Writes one space, decrements the count, and calls itself.

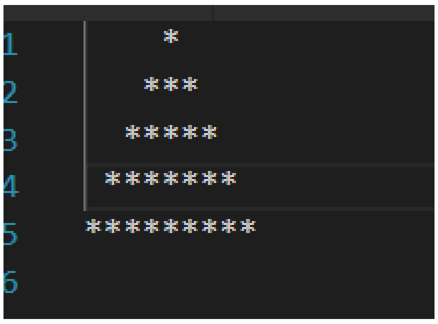
print_stars/2

- **Purpose**: Writes a specified number of * symbols to the file stream.
    - Base Case: Stops when there are no stars to write.
    - Recursive Case: Writes one *, decrements the count, and calls itself.

Testing:



```
4 ?- isosceles_triangle_pattern_file(5, 'triangle.txt').
Isosceles triangle pattern written to file: triangle.txt
true .
```



```
    *
   ***
  *****
 *******
*********

```

Task B： Parsing Game Character Descriptions with Definite Clause Grammars (DCGs)

1. DCG Rule: character_description

- **Purpose:** Parses a character description and validates each component. If all validations pass, it asserts the character into the knowledge base using the assert_character/7 predicate.
- **Components:**
    - **Input format:** [Type, Subtype, Sequence, HealthLevel, Weapon, MovementStyle]
    - **Validation:** Calls validation predicates (validate_*) to ensure that:
        - The type and subtype are appropriate for the character.
        - Health level, weapon type, and movement style are valid.
        - Movement direction is determined logically for the character type and weapon possession.
    - **Asserting:** Stores the validated character in the knowledge base using assert/1.

2. Movement Direction: determine_movement_direction/3

- **Purpose:** Determines the movement direction of the character based on its type and weapon possession.
- **Rules:**
    - Enemies always move towards.
    - Heroes with weapons move towards, while those without weapons move away.

3. Validation Predicates

1. **validate_type/1**
    - **Purpose:** Ensures that the character type is either enemy or hero.
    - **Logic:** Uses member/2 to check membership in the valid types list.
2. **validate_subtype/2**
    - **Purpose:** Ensures the subtype matches the character type.
    - **Logic:** Checks that:
        - Enemies have subtypes: darkwizard, demon, or basilisk.
        - Heroes have subtypes: wizard, mage, or elf.
3. **validate_health/1**
    - **Purpose:** Ensures the health level is valid.
    - **Logic:** Checks membership in the predefined health levels list.
4. **validate_weapon/2**
    - **Purpose:** Validates weapon possession rules:
        - Heroes can have has_weapon or no_weapon.
        - Enemies always have no_weapon.

validate_movement_style/1

- **Purpose:** Ensures the movement style is one of jerky, stealthy, or smoothly.

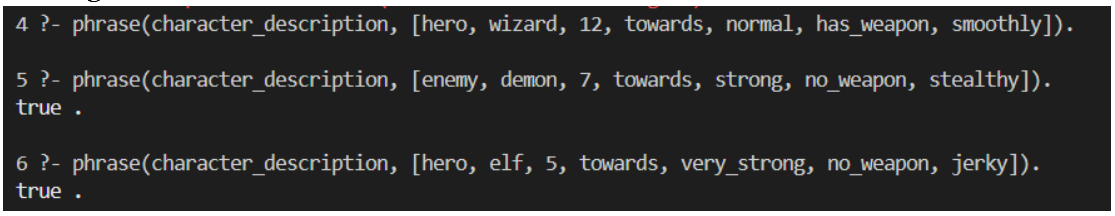
4. Assertion: assert_character/7

- **Purpose:** Inserts a validated character into the dynamic knowledge base.
- **Logic:** Uses assert/1 to store the character with all its attributes.

5. Retrieval: get_character/7

- **Purpose:** Retrieves a character from the knowledge base by matching its attributes.
- **Logic:** Uses the dynamic character/7 predicate to retrieve stored characters.

Testing:



```
4 ?- phrase(character_description, [hero, wizard, 12, towards, normal, has_weapon, smoothly]).

5 ?- phrase(character_description, [enemy, demon, 7, towards, strong, no_weapon, stealthy]).
true .

6 ?- phrase(character_description, [hero, elf, 5, towards, very_strong, no_weapon, jerky]).
true .
```

Task C: Library Management System with Prolog

1. Dynamic Predicates

- **book/4**: Represents books in the library with attributes: Title, Author, Year, and Genre.
- **borrowed/4**: Tracks books that have been borrowed with the same attributes.

*Both predicates are declared dynamic to allow runtime modifications.

2. Core Operations

1. **add_book/4**
   - **Purpose**: Adds a book to the library if it doesn't already exist.
   - **Logic**: Ensures that the book is not already present before adding it.
2. **remove_book/4**
   - **Purpose**: Removes a book from the library.
   - **Logic**: Deletes the book's entry from the knowledge base.
3. **is_available/4**
   - **Purpose**: Checks if a book is available for borrowing.
   - **Logic**: Ensures the book exists and has not been marked as borrowed.
4. **borrow_book/4**
   - **Purpose**: Marks a book as borrowed if it is available.
   - **Logic**: Ensures the book can be borrowed and adds it to the borrowed list.
5. **return_book/4**
   - **Purpose**: Marks a borrowed book as returned.
   - **Logic**: Removes the book from the borrowed list.

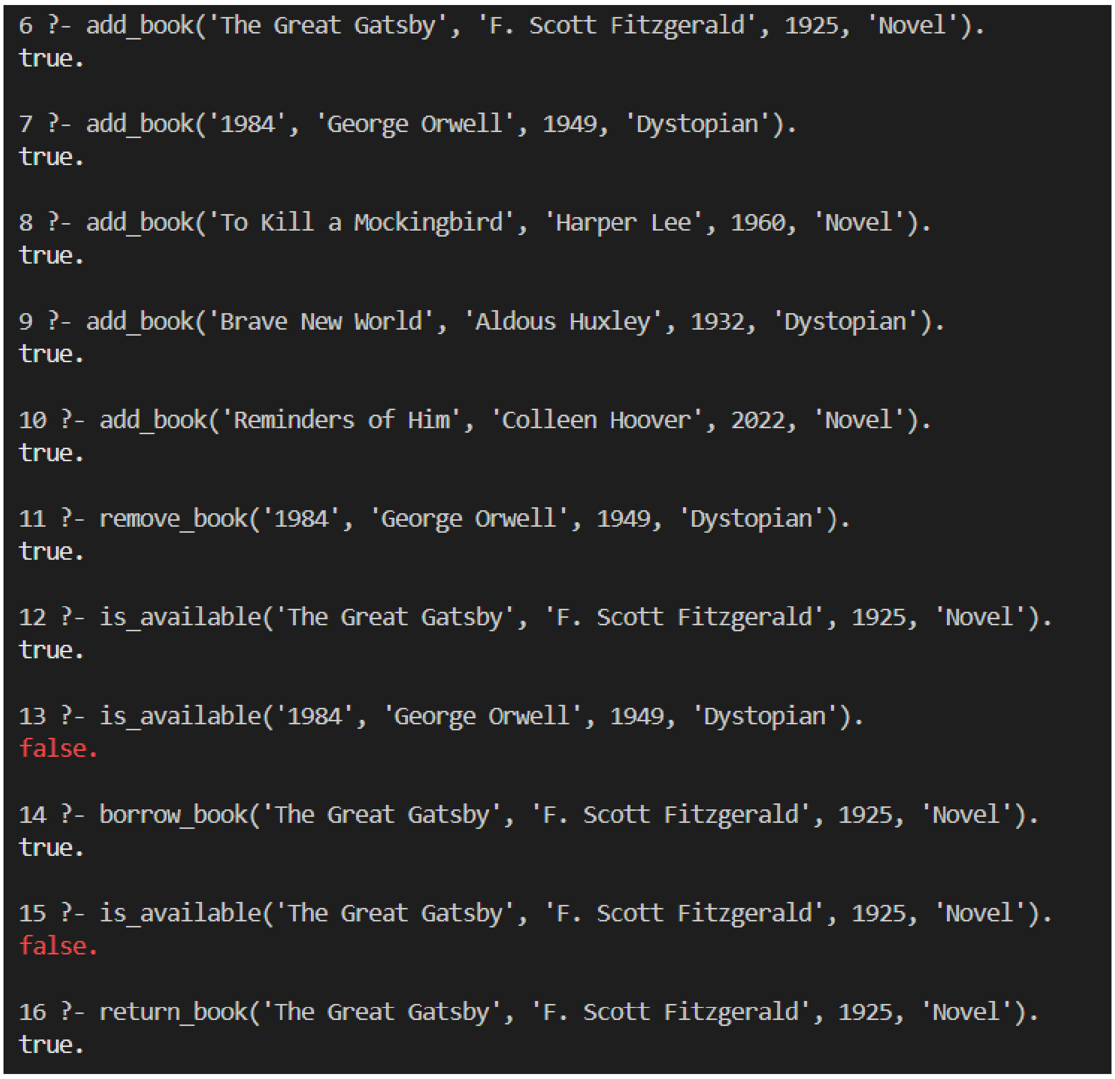
3. Search Operations

1. **find_by_author/2**
   - **Purpose**: Finds all books by a specific author.
   - **Logic**: Collects all titles associated with the specified author.
2. **find_by_genre/2**
   - **Purpose**: Finds all books of a specific genre.
   - **Logic**: Collects all titles associated with the specified genre.
3. **find_by_year/2**
   - **Purpose**: Finds all books published in a specific year.
   - **Logic**: Collects all titles associated with the specified year.

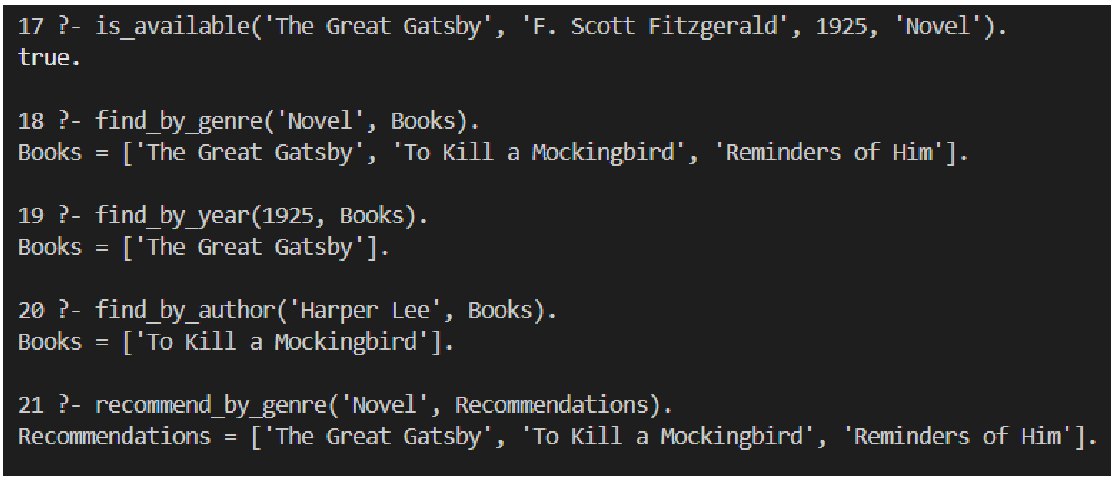
4. Recommendations

1. **recommend_by_genre/2**
   - **Purpose**: Recommends books of a specific genre.
   - **Logic**: Suggests all titles in the specified genre.
2. **recommend_by_author/2**
   - **Purpose**: Recommends books by a specific author.
   - **Logic**: Suggests all titles by the specified author.

Testing:



```
6 ?- add_book('The Great Gatsby', 'F. Scott Fitzgerald', 1925, 'Novel').
true.

7 ?- add_book('1984', 'George Orwell', 1949, 'Dystopian').
true.

8 ?- add_book('To Kill a Mockingbird', 'Harper Lee', 1960, 'Novel').
true.

9 ?- add_book('Brave New World', 'Aldous Huxley', 1932, 'Dystopian').
true.

10 ?- add_book('Reminders of Him', 'Colleen Hoover', 2022, 'Novel').
true.

11 ?- remove_book('1984', 'George Orwell', 1949, 'Dystopian').
true.

12 ?- is_available('The Great Gatsby', 'F. Scott Fitzgerald', 1925, 'Novel').
true.

13 ?- is_available('1984', 'George Orwell', 1949, 'Dystopian').
false.

14 ?- borrow_book('The Great Gatsby', 'F. Scott Fitzgerald', 1925, 'Novel').
true.

15 ?- is_available('The Great Gatsby', 'F. Scott Fitzgerald', 1925, 'Novel').
false.

16 ?- return_book('The Great Gatsby', 'F. Scott Fitzgerald', 1925, 'Novel').
true.
```



```
17 ?- is_available('The Great Gatsby', 'F. Scott Fitzgerald', 1925, 'Novel').
true.

18 ?- find_by_genre('Novel', Books).
Books = ['The Great Gatsby', 'To Kill a Mockingbird', 'Reminders of Him'].

19 ?- find_by_year(1925, Books).
Books = ['The Great Gatsby'].

20 ?- find_by_author('Harper Lee', Books).
Books = ['To Kill a Mockingbird'].

21 ?- recommend_by_genre('Novel', Recommendations).
Recommendations = ['The Great Gatsby', 'To Kill a Mockingbird', 'Reminders of Him'].
```