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).

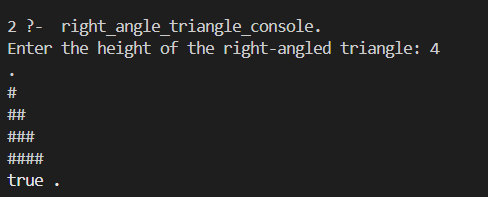
**print\_triangle/2**

* **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.
* The predicate checks if Current =< Height and continues printing rows. If Current > Height, the recursion ends.

**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:**



**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.

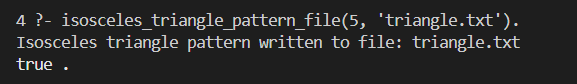
**print\_spaces/2**

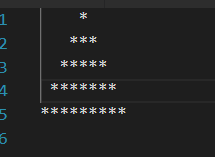
* **Purpose**: Writes a specified number of spaces to the file stream.
  + **Base Case**: Stops when there are no spaces to write.
  + **Recursive** **Case**: Writes one space, decrements the count, and calls itself.

**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:**





**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.

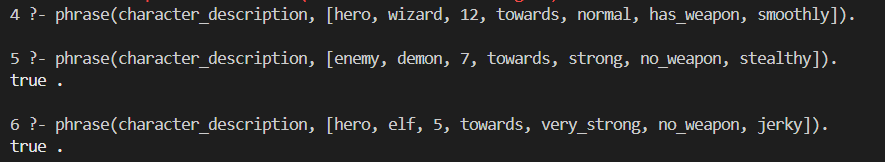
**4. Assertion: assert\_character/7**

* **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:**



**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:**

