Name:  
Date:  
Final IDE Program Share Link:

**PART 1: Defining Your Problem**

**Task:** Create a program that generates a unique, random short story each time it's run. The story will be based on user-selected themes or random prompts.

**Requirements:**

* Problem: Generate a random short story using pre-written sentences..
* Input: User selects a theme (e.g., Adventure, Mystery, Comedy) or lets the program pick randomly.
* Program Functionality: The program selects random story elements like characters, settings, conflicts, and resolutions based on the chosen theme.
* Output: A complete, humorous, or intriguing short story.

**PART 2: Working Through Specific Examples**

**Task:** Write down clear steps to solve a simple version of your problem.

**Requirements:**

**Example 1**: User selects "Adventure"

1. Input: User chooses "Adventure."
2. Steps:
   1. Randomly select a character (e.g., "a daring astronaut").
   2. Randomly select a setting (e.g., "on a distant planet").
   3. Randomly select a conflict (e.g., "encounters an alien civilization").
   4. Randomly select a resolution (e.g., "becomes their ambassador to Earth").
3. Output: "An astronaut, lost on a distant planet, encounters an alien civilization and unexpectedly becomes their ambassador to Earth."

**Example 2**: User selects "Comedy"

1. Input: User chooses "Comedy."
2. Steps:
   1. Randomly select a character (e.g., "a clumsy detective").
   2. Randomly select a setting (e.g., "at a fancy dinner party").
   3. Randomly select a conflict (e.g., "accidentally reveals the host's biggest secret").
   4. Randomly select a resolution (e.g., "ends up being the life of the party").
3. Output: "A clumsy detective at a fancy dinner party accidentally reveals the host's biggest secret, but somehow ends up being the life of the party."

**PART 3: Generalizing Into Pseudocode**

**Task:** Write out the general sequence your program will use.

**Requirements:**

* Write pseudocode for full functionality.
* Include logical steps (variables, loops, conditionals).

### ****Pseudocode: Random Story Generator****

**Initialize Themes**

* 1. Create a dictionary to store themes:
     1. Adventure: list of characters, settings, conflicts, and resolutions.
     2. Comedy: list of characters, settings, conflicts, and resolutions.
     3. Mystery: list of characters, settings, conflicts, and resolutions.

**Ask for User Input**

* 1. Prompt the user to select a theme (Adventure, Comedy, Mystery) or choose "Random."

**Select Theme**

* 1. If the user chooses a specific theme:
     1. Use the corresponding list of story elements.
  2. If the user chooses "Random":
     1. Randomly select a theme from the dictionary.

**Generate Random Story Elements**

* 1. Randomly select one character, one setting, one conflict, and one resolution from the chosen theme.

**Assemble the Story**

* 1. Combine the selected elements into a structured short story.
  2. Example format:  
     "Once upon a time, [character] was [setting]. They encountered [conflict], and finally, [resolution]."

**Display the Story**

* 1. Print the completed story for the user.

**Optional: Ask for Replay**

* 1. Prompt the user to generate another story or exit the program.

### Example Themes for Implementation

#### Adventure

* **Characters**: A daring astronaut, a fearless explorer, a curious archaeologist.
* **Settings**: On a distant planet, in a hidden jungle, inside an ancient pyramid.
* **Conflicts**: Encounters an alien civilization, discovers a lost artifact, faces a giant creature.
* **Resolutions**: Becomes their ambassador, uncovers the artifact's secrets, defeats the creature.

#### Comedy

* **Characters**: A clumsy detective, a quirky magician, a forgetful professor.
* **Settings**: At a fancy dinner party, during a live TV show, in a chaotic classroom.
* **Conflicts**: Accidentally reveals the host's secret, loses their magic wand, trips during a presentation.
* **Resolutions**: Becomes the life of the party, improvises with surprising success, turns the mishap into a joke.

#### Mystery

* **Characters**: A brilliant detective, an amateur sleuth, a reclusive writer.
* **Settings**: In a haunted mansion, at a quiet seaside town, during a high-profile gala.
* **Conflicts**: Unravels a series of strange clues, encounters an unexpected suspect, finds themselves under suspicion.
* **Resolutions**: Solves the case dramatically, exposes the real culprit, clears their name.

**PART 4: Testing Your Program**

**Task:** Describe your tests, record errors, and explain how you fixed them.

**Requirements:**

Test Case 1:

Input: User selects "Adventure" as the theme.

Expected Output: A complete story with adventure-themed elements.

Result: Confirm the generated story contains appropriate characters, settings, conflicts, and resolutions for the Adventure theme.

Test Case 2:

Input: User selects "Random" as the theme.

Expected Output: A complete story with elements chosen from a random theme.

Result: Verify that the program randomly selects a theme and generates a coherent story.

Test Case 3:

Input: User provides invalid input (e.g., "Sci-Fi").

Expected Output: Program prompts the user to re-enter a valid choice.

Result: Confirm the error handling correctly re-prompts the user without crashing.

Error Handling:

Error: User inputs invalid theme choice (e.g., misspelling).

Fix: Added a loop to re-prompt for valid input.

Error: Story elements not combining correctly due to missing or mismatched keys in the themes dictionary.

Fix: Validated that all themes have properly aligned keys and values before running the program.

Error: Empty output when random.choice() fails (e.g., empty list).

Fix: Ensured all lists in the themes dictionary are populated and added fallback default values for each category.

**PART 5: Commenting Your Program**

**Task:** Submit your full program code, including comments.

**Requirements:**

* Include comments explaining each part of the code.

**PART 6: Your Completed Program**

**Task:** Provide the IDE share link to your full program code.

**Requirements:**

* Ensure the program works correctly with comments included.