**Add New City**

**Purpose:** The purpose of this function is to **add a new city to the city list**.

**Prototype:** void addCity(char cities[][50], int\*count);

**Parameters:**

* cities[][50] – A 2D character array used to store city names.
* \*count – A pointer to the current number of cities in the list.

**How It Works:**

1. **Check capacity**
   * First, it checks if the city list is full (\*count >= MAX\_CITIES).
   * If full, it displays the message: “City list is full!”.
2. **Input city name**
   * It takes the city name from the user and stores it in the name array.
3. **Check uniqueness**
   * Compares the new name with existing cities in the list.
   * If any character does not match, notUnique = 0.
   * If the name matches an existing city, unique = 0.
4. **Add city**
   * If the name is unique, it copies the name into cities[\*count].
   * Increments the city count (\*count)++.
   * Displays: “City added successfully!”.
5. **Duplicate case**
   * If the name already exists in the list, it displays: “City name already exists!”.

A computer screen shot of a black screen

Description automatically generated

**Display all cities**

**Purpose:**  
The purpose of this function is to **display all the cities currently stored in the city list** in a numbered format.

**Prototype:**

void displayCities(char cities[][50], int count);

**Parameters:**

* cities[][50] – A 2D character array containing city names.
* count – The current number of cities in the list.

**How It Works:**

1. **Check if list is empty**
   * If count is 0, it displays: “No cities to display.”
2. **Display cities**
   * Iterates through all cities from index 0 to count - 1.
   * Prints each city with a number (1, 2, 3, …) for clarity.
   * Uses a while loop with putchar to print each character of the city name.

**Example Output:**

A computer screen with a black screen

Description automatically generated

### ****Function: renameCity****

**Purpose:**  
The purpose of this function is to **rename an existing city in the city list** while ensuring the new name is unique.

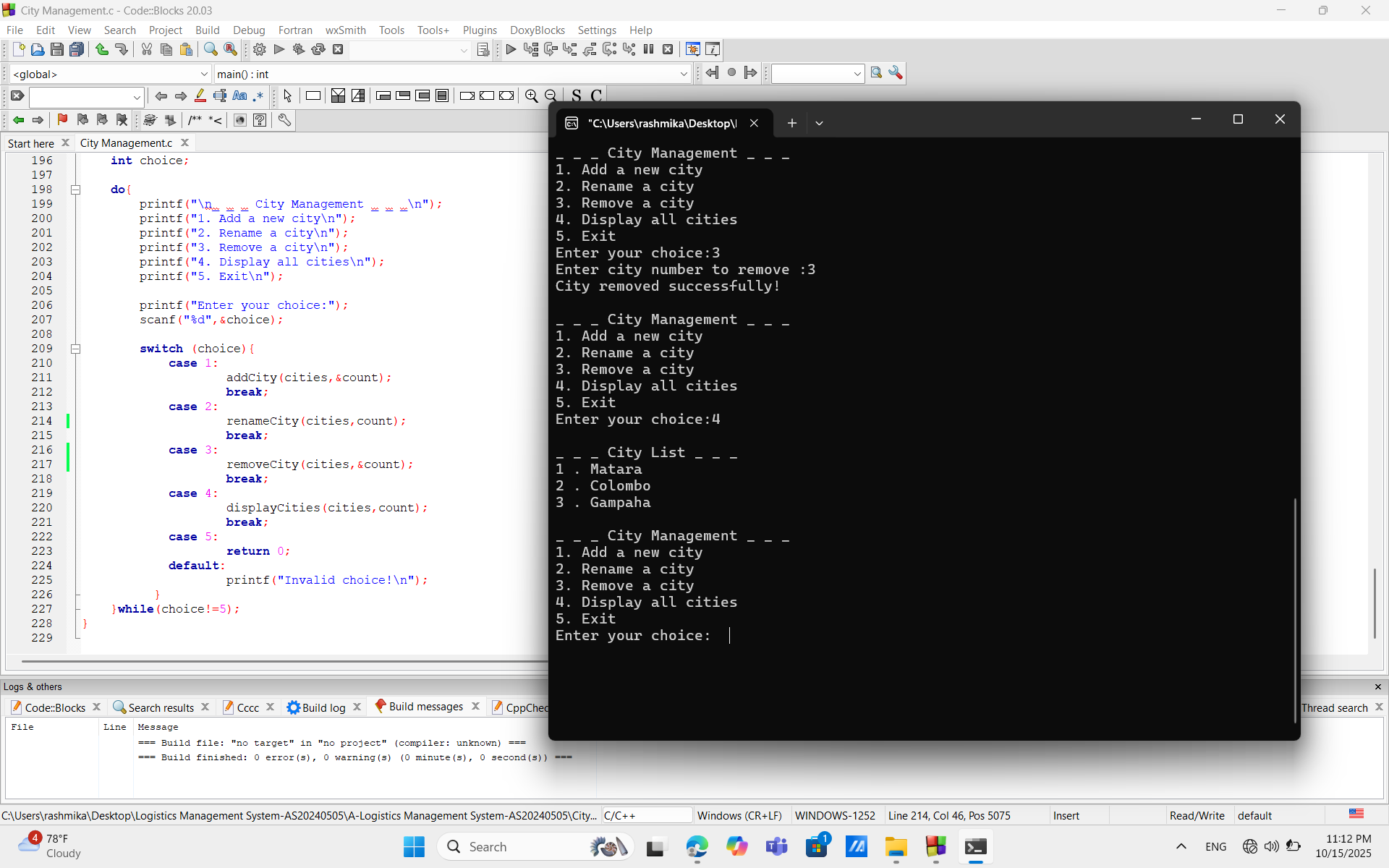
**Prototype:**

void renameCity(char cities[][50], int count);

**Parameters:**

* cities[][50] – A 2D character array containing the city names.
* count – The current number of cities in the list.

**How It Works:**

1. **Check if list is empty**
   * If count is 0, it displays: “No cities to rename.”
2. **Select city to rename**
   * Prompts the user to enter the city number.
   * If the number is invalid (less than 1 or greater than count), it displays: “Invalid city number.”
3. **Input new name**
   * Takes the new city name from the user into newName.
4. **Check uniqueness**
   * Compares newName with all existing cities.
   * If a duplicate is found, unique = 0.
5. **Rename city**
   * If the name is unique, copies newName into the selected city’s array (cities[cityNumber-1]).
   * Displays: “City renamed successfully!”
6. **Duplicate case**
   * If the new name already exists, displays: “City name already exists!”

### ****Function: removeCity****

**Purpose:**  
The purpose of this function is to **remove an existing city from the city list** and adjust the list accordingly.

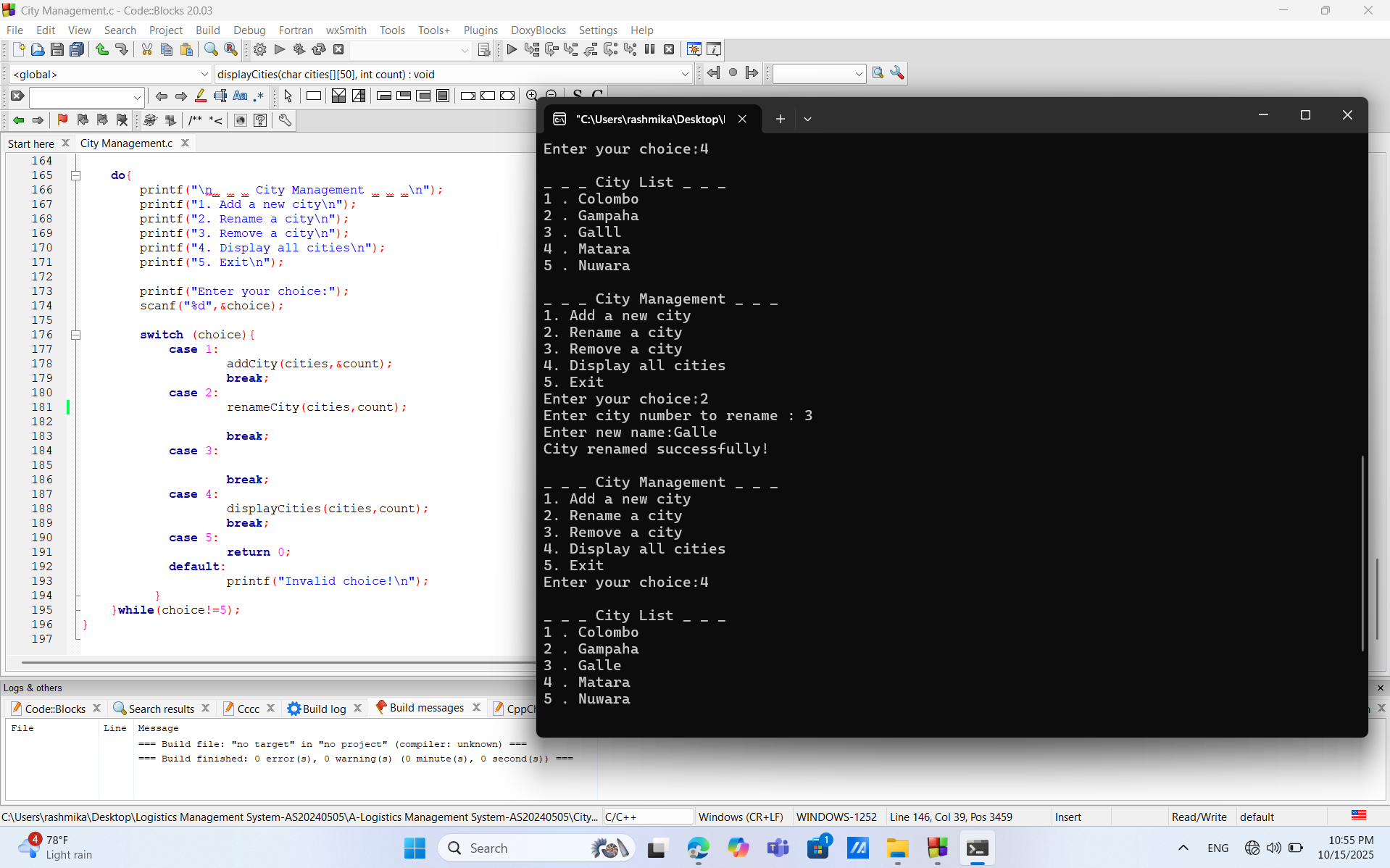
**Prototype:**

void removeCity(char cities[][50], int \*count);

**Parameters:**

* cities[][50] – A 2D character array containing the city names.
* \*count – A pointer to the current number of cities in the list.

**How It Works:**

1. **Check if list is empty**
   * If \*count is 0, it displays: “No cities to remove.”
2. **Select city to remove**
   * Prompts the user to enter the city number.
   * If the number is invalid (less than 1 or greater than \*count), it displays: “Invalid city number.”
3. **Remove city**
   * Shifts all cities after the selected city one position up in the array to fill the gap.
   * Decrements the city count ((\*count)--).
4. **Confirmation**
   * Displays: “City removed successfully!”