CSE3150 Homework 2: Greeting Formatter

Homework

1. **Project Structure:** Organize your files into two directories:
   - `include/`: Stores your header file (`.h`).
   - `src/`: Stores your C++ source files (`.cpp`).
2. **GreetingUtils Namespace:** Create a namespace called `GreetingUtils`. Its function declarations should be in `include/greeting_utils.h` and definitions in `src/greeting_utils.cpp`.
   - Remember to use **header guards** in `greeting_utils.h`!
   - Create a function: `std::string create_message(const std::string& name);`. This function should take a person's name and return a greeting string, for example, "Hello, `<name>`!".
   - Create a second function: `char* format_as_c_string(const std::string& msg);`. This function must:
     (a) Take a constant string reference as input.
     (b) **Dynamically allocate a `char` array on the heap** that is large enough to hold the entire message, plus a null terminator.
     (c) Copy the characters from the `std::string` into the new `char` array.
     (d) Add the null terminator `'\0'` at the end of the `char` array.
     (e) Return the **pointer** to the new heap-allocated `char` array.
3. **main.cpp:** In `src/main.cpp`, write a program that:
   - Prompts the user to enter their name using `std::cout`.
   - Reads the full line of input using `std::getline`.
   - Calls your `GreetingUtils::create_message` function to generate the greeting.
   - Passes that greeting to your `GreetingUtils::format_as_c_string` function to get the dynamically allocated C-style string.
   - Prints the greeting message to the console from the returned `char*` pointer.
   - **CRITICAL:** Frees the heap-allocated memory using `delete[]`.
4. **Submission:**
   - Create a new public GitHub repository named `cse3150_hw_2`.
   - Add a `.gitignore` file to exclude your executable (`greeter`) and any object files (`*.o`).
   - Push your organized code (`include/`, `src/`).

Testing with Pytest

You can use the following test file to check your code

```python
import subprocess
import pytest
import os

def run_greeter(input_text):
    """Helper function to run the compiled C++ greeter with input."""

    # The input_text needs a newline, as if the user pressed Enter
    input_with_newline = input_text + "\n"

    result = subprocess.run(
        ["./greeter"],
        input=input_with_newline,
        capture_output=True,
        text=True,
        check=True
    )
    # The output from the C++ program includes the prompt and the final message.
    lines = result.stdout.strip().splitlines()
    return lines[-1] if lines else ""

def test_greeting_with_spaces():
    """Test the greeting with a name that includes spaces."""
    name = "First Last"
    expected_output = f"Hello, {name}!"
    actual_output = run_greeter(name)
    assert actual_output == expected_output
```