CSE3150 Week 2 Lab: String Parser and Dynamic Memory

Dr. Justin furuness

Lab

1. **Project Structure:** Organize your files into two directories:
   - `include/`: Stores your header file (`.h`).
   - `src/`: Stores your C++ source files (`.cpp`).
2. **StringUtils Namespace:** Create a namespace called `StringUtils`. Its function declarations should be in `include/parser.h` and definitions in `src/parser.cpp`.
   - Remember to use **header guards** in `parser.h`!
   - Create a function: `void parseName(const std::string& fullName, std::string* firstName, std::string* lastName);`. This function should take a full name and use the **pointer** arguments to return the separated first and last names.
   - Create a second function: `std::string getUsername(const std::string& email);`. This function should take a full email address and return the username part (everything before the `@` symbol).
3. **main.cpp:** In `src/main.cpp`, write a program that:
   - Prompts the user to enter their full name and email address on separate lines.
   - Reads the full line of input for the name using `std::getline`.
   - **Dynamically allocates two `std::string` objects on the heap** using `new`. These will be used to store the first and last names.
   - Calls your `StringUtils::parseName` function, passing the pointers to your newly allocated strings.
   - Calls your `StringUtils::getUsername` function to get the username.
   - Prints the parsed first name, last name, and username to the console. You will need to **dereference** the pointers to print the names.
   - Frees the heap-allocated memory for the first and last name strings using `delete`. No memory leaks!!
4. **Submission:**
   - Create a new public GitHub repository named `cse3150_week_2_lab`.
   - Add a `.gitignore` file to exclude your executable (`parser_app`) and any object files (`*.o`).
   - Push your 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_parser(full_name, email):
    """Helper function to run the C++ parser with input."""

    # Combine inputs with newlines, as if the user pressed Enter after each
    input_text = f"{full_name}\n{email}\n"

    result = subprocess.run(
        ["./parser_app"],
        input=input_text,
        capture_output=True,
        text=True,
        check=True
    )

    # Parse the output to find the key-value pairs
    output_data = {}
    for line in result.stdout.strip().splitlines():
        if ":" in line:
            key, value = line.split(":", 1)
            output_data[key.strip()] = value.strip()

    return output_data

def test_simple_name_and_email():
    """Tests a standard first and last name."""
    name = "Jane Doe"
    email = "jane.doe@example.com"

    parsed_info = run_parser(name, email)

    assert parsed_info.get("First Name") == "Jane"
    assert parsed_info.get("Last Name") == "Doe"
    assert parsed_info.get("Username") == "jane.doe"

def test_name_with_middle_initial():
    """Tests a name that includes a middle initial."""
    name = "John Doe"
    email = "john.doe@uconn.edu"

    parsed_info = run_parser(name, email)

    assert parsed_info.get("First Name") == "John"
    assert parsed_info.get("Last Name") == "Doe"
    assert parsed_info.get("Username") == "john.doe"
```