**Program 11:String class with copy constructor and overloaded assignment operator**

#include <iostream>

#include <cstring>

class String {

private:

char\* str; // Pointer to the character array

int length; // Length of the string

public:

// Default constructor

String() : str(nullptr), length(0) {}

// Parameterized constructor

String(const char\* s) {

length = std::strlen(s);

str = new char[length + 1];

std::strcpy(str, s);

}

// Copy constructor (Deep copy)

String(const String& other) {

length = other.length;

str = new char[length + 1];

std::strcpy(str, other.str);

}

// Overload assignment operator (Deep copy)

String& operator=(const String& other) {

// Check for self-assignment

if (this == &other) {

return \*this;

}

// Deallocate the existing memory

delete[] str;

// Perform deep copy

length = other.length;

str = new char[length + 1];

std::strcpy(str, other.str);

return \*this;

}

// Destructor

~String() {

delete[] str;

}

// Getter function to retrieve the string

const char\* c\_str() const {

return str;

}

// Getter function to retrieve the length of the string

int size() const {

return length;

}

};

int main() {

// Create a String object

String s1("Hello, World!");

// Use the copy constructor to make a deep copy

String s2(s1);

// Use the assignment operator to make another deep copy

String s3 = s1;

// Display the strings

std::cout << "s1: " << s1.c\_str() << ", Length: " << s1.size() << std::endl;

std::cout << "s2: " << s2.c\_str() << ", Length: " << s2.size() << std::endl;

std::cout << "s3: " << s3.c\_str() << ", Length: " << s3.size() << std::endl;

return 0;

}

**Output Snip:**