Offline 3: Operator Overloading

Write a C++ program that should use operator overloading to provide intuitive ways of interacting with a collection of planets, their details, and perform operations between them. Complete the “Planet” and “SolarSystem” classes with only necessary operator overloading functions so that the main function can produce correct output.

Assumption f**or behavior of `[]` operator with name lookup**: If the planet name doesn't match any planet in the solar system, the first planet (`planets[0]`) is returned. For example, if we call **solarSystem["Venus"]** and there exist no such planet in the solar system, `planets[0]` is returned.

```
#include <iostream>
#include <cstring>

using namespace std;

class Planet {
   char* name;    // Pointer for the name of the planet
   int distance;    // Distance from the sun in millions of kilometers

public:
   Planet() {
      name = new char[1];
      name[0] = '\0';
      distance = 0;
   }

   Planet(const char* planetName, int dist) {
      name = new char[strlen(planetName) + 1];
      strcpy(name, planetName);
      distance = dist;
   }

   Planet(const Planet& other) {
      name = new char[strlen(other.name) + 1];
      strcpy(name, other.name);
      distance = other.distance;
   }

 ~Planet() {
      delete[] name;
   }

   void display() {
      cout << "Planet: " << name << ", Distance: " << distance << " million km\n";
   }
```

```
// Method to get the planet name
const char* getName(){
        return name;
    }
};

class SolarSystem {
    Planet planets[10];
    int planetCount;
public:
    SolarSystem() {
        planetCount = 0;
    }

    // Add a planet to the solar system
    void addPlanet(const char* name, int distance) {
        if (planetCount < 10) {
            Planet p(name, distance);
            planets[planetCount] = p;
            planetCount++;
        } else {
            cout << "Solar system is full. Can't add more planets.\n";
        }
    }
};

int main() {
    SolarSystem solarSystem;

    // Add planets to the solar system
    solarSystem.addPlanet("Invalid", -1);
    solarSystem.addPlanet("Sun", 0);
    solarSystem.addPlanet("Earth", 150);
    solarSystem.addPlanet("Mars", 228);

    // Display all planets in the solar system
    solarSystem[1].display();
    solarSystem[2].display();
    solarSystem[3].display();

    solarSystem[3] = 140 + solarSystem[3];
    solarSystem[3].display();
    Planet p = solarSystem["Earth"] + solarSystem["Mars"];
    p.display();

    if (solarSystem[2] < solarSystem[3]) {
        cout << "Earth is closer to the Sun than Mars\n";
    } else {
        cout << "Mars is closer to the Sun than Earth\n";
    }

    p = solarSystem[1];
    p.display();

    return 0;
}
```

Output:

```
Planet: Sun, Distance: 0 million km
Planet: Earth, Distance: 150 million km
Planet: Mars, Distance: 228 million km
Planet: Mars, Distance: 368 million km
Planet: Combined, Distance: 518 million km
Earth is closer to the Sun than Mars
Planet: Sun, Distance: 0 million km
```

Submission guidelines:

1. Create a folder named after your student ID.
2. Write your code in a single file named your_ID.cpp.
3. Move the .cpp file into the folder from step 1.
4. Zip the folder and name it your student ID. Submit this zip on Moodle.

For example,

```
1805010/
  └── 1805010.cpp
```

You would then submit 1805010.zip

Deadline: October 27 (11:55 PM)

Please do not copy from any source (friends, internet, AI tools like ChatGPT, etc.). Doing so may lead to severe penalties.