This is part 2 for Exam 2. This part is worth 30 points (part 1 is implemented as a blackboard exam and is worth 70 points).

Please read the instructions carefully.

Using the `Zorb` class, create a derived class `KiloZorb`. You may not add any new data or function members to the derived class. A `KiloZorb` differs from a `Zorb` in that a `KiloZorb` has 1000 times the power of a `Zorb`.

1. Show on the listing of the `Zorb` class show how the base class would need to be modified such that a derived class could override the definitions of any member functions.
2. Write the class declaration for the `KiloZorb` class, derived from the `Zorb` base class. Within your class declaration, show the implementation of the `KiloZorb` value constructor as an inline function, implemented in terms of the base class `Zorb` constructor. Recall that the new `KiloZorb` will have 1000 the power indicated by the power parameter.
3. Within your class declaration, show the implementation of `operator<` for `KiloZorb` as an inline function, implemented in terms of the base class `Zorb` version of `operator<`.
4. Within your class declaration, show the implementation of `operator+` for `KiloZorb` as an inline function, implemented in terms of the base class `Zorb` version of `operator+`. The combined `KiloZorb` will have 1000 times the sum of the powers of the original `Zorbs`.

```
// The Zorb game - the base class
#include <iostream>
#include <string>
using namespace std;
class Zorb {
    public:
        // value constructor
        Zorb(int p, int t) : _power(p), _team_id(t) {}
        bool operator< (const Zorb&) const;
        Zorb operator+ (const Zorb&) const;
        int getPower() { return _power; }
        int getTeamID() { return _team_id; }
    friend ostream& operator<< (ostream&, const Zorb&);
    private:
        int _power;
        int _team_id;
};
// operator< usage: Zorb z1, z2; if (z1 < z2) {}
// In the expression z1<z2, "this" Zorb is z1 (the left operand).
// Returns true if "this" Zorb (left operand) has a different team ID
// AND greater power than the other Zorb (right operand).
bool Zorb::operator< (const Zorb& z) const {
    if (this->_team_id == z._team_id)
        return false;
    else
        return this->_power < z._power;
}

// operator+ usage: Zorb z3 = z1 + z2;
// Constructs a new Zorb (here assigned to z3) with the combined power
// of the original two Zorbs (z1 and z2), and the team ID of the stronger of
// the original two Zorbs.
// In the expression z1+z2, "this" Zorb is z1 (the left operand).
Zorb Zorb::operator+ (const Zorb& z) const {
    return Zorb(this->_power + z._power,
                ((*this < z) ? z._team_id : this->_team_id) );
}
```