```cpp

//Sartaj Khan Midterm
//Date: 26/04/23


#include "PrefixString.h"
#include <cassert>


PrefixString::PrefixString(char aExtension) noexcept :
    fCode(static_cast<uint16_t>(-1)),
    fPrefix(static_cast<uint16_t>(-1)),
    fExtension(aExtension)
{}

PrefixString::PrefixString(uint16_t aPrefix, char aExtension) noexcept :
    fCode(static_cast<uint16_t>(-1)),
    fPrefix(aPrefix),
    fExtension(aExtension)
{}

uint16_t PrefixString::getCode() const noexcept {
    return fCode;
}

void PrefixString::setCode(uint16_t aCode) noexcept {
    fCode = aCode;
}

uint16_t PrefixString::w() const noexcept {
    return fPrefix;
}

char PrefixString::K() const noexcept {
    return fExtension;
}

PrefixString PrefixString::operator+(char aExtension) const noexcept {
    assert(fCode != -1);
    return PrefixString(fCode, aExtension);
}

bool PrefixString::operator==(const PrefixString& aOther) const noexcept {
    return fPrefix == aOther.w() && fExtension == aOther.K();
}

std::ostream& operator<<(std::ostream& aOStream, const PrefixString& aObject) {
    return aOStream << "(" << aObject.getCode() << "," << aObject.w() << "," << aObject.K() << ")";
}
```