```
#include <windows.h>
#include <string>
#include "packet.h"
#include "crc.h"
#include "utils.h"

using namespace std;

Packet::Packet(): packet_(""), length_(0) {}

Packet::Packet(const Packet& packet): packet_(packet.packet_), length_(packet.length_)
{}  // copy constructor

Packet::Packet(std::string data): packet_(data), length_(data.length() + 4) {
    // turn string into packet
    packet_.insert(0, 1, SOH);
    packet_.insert(1, 1, (char)data.length() + 4);
    packet_.insert(2, 1, NONE);
}

Packet::Packet(char flag): length_(minLength_) {
    packet_.insert(0, 1, SOH);
    packet_.insert(1, 1, (char)0x04);
    packet_.insert(2, 1, flag);
}

void Packet::seq(bool toggle) { // only call once on a packet!!!
    packet_[2] = (toggle) ? (packet_[2] | SEQ) : (packet_[2] & (!SEQ));     // set SEQ
bit
}

bool Packet::seq() {
    return (packet_[2] & SEQ) == SEQ;
}

void Packet::append(char c) {
    packet_.append(1, c);
}

bool Packet::complete() {
    return packet_.length() >= minLength_ && packet_.length() == packet_[1];
}

bool Packet::valid() {
    ENSURE_BOOL(packet_.length() >= minLength_);
    ENSURE_BOOL(packet_[0] == SOH);
    ENSURE_BOOL(packet_[1] == packet_.length());
    ENSURE_BOOL(CRC::calc(packet_.substr(0, packet_.length()-2)));
    return true;
}

int Packet::flags() {
    return (packet_[2] & (!SEQ));
}

string Packet::data() {
    return packet_.substr(4, packet_.length()-2);
}

void Packet::calcCRC() {
    if (packet_.length() == length_) {  // remove CRC
        packet_.erase(packet_.end());
    }
    // re-calculate CRC
    //packet_ += CRC::calc(packet_);    // re-calculate CRC
    packet_.insert(packet_.end(), 1, CRC::calc(packet_));
}

string Packet::toString() {
    calcCRC();
    // return packet
    return packet_;
}

void Packet::clear() {
    length_ = 0;
    packet_.clear();
}

int Packet::length() {
    return (int)length_;
}

bool Packet::cmd() {
    return length_ == minLength_;
}
```