Задача: Нека имаме следния протокол (формат) за изпращане и получаване на съобщения:

- всяко съобщение получава уникално ID при създаването си.
- също така, в зависимост от мрежата, по която ще се изпращат съобщенията, има ограничение спрямо максималната дължина на едно съобщение (като в тази дължина не влизат ID-то и останалата системна информация на съобщението).

HINT 1: Едно съобщение може да е с **произволна дължина**, но има ограничение при изпращането - тоест, ще трябва да разбием съобщението на няколко части, когато го изпращаме. В статични променливи на класа ще пазим:

- последното използвано ID (започваме от 0 и на всяка нова инстанция на класа ще увеличаваме)
- максималната възможна дължина за изпращане по мрежата.

Едно съобщение, според нашия протокол, съдържа следната информация:

- **unsigned int id** - уникално ID, получавано при създаване на съобщението
- **unsigned int previousMessageId** - ако съобщението е по-дълго от максимално позволената дължина, то трябва да бъде разбито на няколко части при изпращането (извикване на метод send()). Тоест, previousMessageId пази id-то на съобщението, което съдържа предишната част
- **bool hasNextPart** - дали има следваща част от съобщението или това е последната част
- **unsigned int payloadLength** - дължина на съобщението (без мета данните - id, previousMessageId, hasNextPart и payloadLength)
- **string payload** - самото съобщение

Релизирайте класа **Message** според описанието по-горе както и следните методи:

- **void send(ostream& os)** - отпечатва съобщението на подадения ostream обект. Ако съобщението е по-дълго от максимално позволената дължина, то трябва да бъде разбито според описаните по-горе правила
- **void receive(istream& is)** - прочита съобщение от подадения istream обект. Ако съобщението е разбито на няколко части, то трябва да бъде сглобено в едно цяло съобщение.

Пример: Искаме да изпратим *“Alabala”*, но максималната дължина е 3 символа. Следователно, при изпращане /* send() */ трябва да разбием съобщението на 3 части.
id=1 prevId=INVALID_ID hasNextPart=true payloadLength=3 payload=“Ala”
id=2 prevId=1 hasNextPart=true payloadLength=3 payload=“bal”
id=3 prevId=2 hasNextPart=false payloadLength=1 payload=“a”

HINT 2: За да не използваме магически числа, нека пазим INVALID_ID също като статична променлива. Нека INVALID_ID да е равно на максималната възможна стойност на типа unsigned int (т.е. 0xFFFFFFFF).

Очакваният резултат с този пример е:
1 4294967295 1 3 Ala
2 1 1 3 bal
3 2 0 1 a

Примерни тестове:

```
void testMessageReceive() {
    stringstream input("1 4294967295 1 3 12|\n2 1 1 3 34|\n3 2 1 3 56|\n4 3 0 1 7");

    Message msg;
    msg.receive(input);

    if (msg.getPayload() == "12|34|56|7") {
        cout << "testMessageReceive: PASSED" << endl;
    }
    else {
        cout << "testMessageReceive: FAILED" << endl;
    }
}

// TODO: test all fields
void testMessageSend() {
    stringstream output;
    string testPayload = "12|34|56|7";
    Message msg(testPayload);

    msg.send(output);

    bool testPassed = true;

    string payload;
    unsigned int length = 0;
    bool hasNext = true;

    while (hasNext) {
        unsigned int tempId, tempPrevId;
        string tempPayload;
        output >> tempId >> tempPrevId >> hasNext >> length;
        output.get();
        getline(output, tempPayload);

        payload += tempPayload;

        if (length != tempPayload.size()) {
            testPassed = false;
        }
    }

    if (payload != testPayload) {
        testPassed = false;
    }

    if (testPassed) {
        cout << "testMessageSend: PASSED" << endl;
    }
    else {
        cout << "testMessageSend: FAILED" << endl;
    }
}
```