Да се напише програма, симулираща работа на мрежа от комутатори(рутери)

1. Дефинирайте клас за мрежов пакет:

- атрибут за съдържание - стринг като използвате char*
- дължина на съдържанието - цяло число
- атрибут за IP адрес, от който е изпратен - стринг
- атрибут за IP адрес, към който е изпратен - стринг
- статичен брояч, който се увеличава при всяко създаване на пакет
- атрибут за номер на пакета - число, взима се от статичния брояч
- метод `int validate()` - връща 1 или 0 според това дали атрибутът за дължина съвпада с дължината на съдържанието
- един единствен конструктор, който приема стойности за съдържанието и IP адресите
- деструктур, който освобождава паметта на съдържанието
- всички атрибути не могат да се променят повече след първоначалното им инициализиране
- конструкторът да хвърля изключения когато:
    - подаденото съдържание е празно
    - някой от подадените адреси е `0.0.0.0` или `127.0.0.0`

2. Дефинирайте клас за рутер:

- атрибут за име - стринг
- атрибут за IP адрес - стринг
- **вектор** от референции към рутери, към които е свързан
- **списък** от информация за познати пътища(routing table)
    - информацията за познат път съдържа IP адреса на рутера, неговия индекс във вектора с рутери и броя пакети, които са пратени
- константен статичен член за максималния брой елементи в списъка от пътища
- константен статичен член за максималния брой скокове(hops), които да се правят при търсене на път
- единствен конструктур, който приема аргументи за името и адреса
- метод `void add_router(const Router& router)`, който добавя съсед
- метод `int query_route(const string address, const int hop_count)`, който проверява дали от този рутер може да се стигне до търсения адрес:
    - методът връща 1 ако този адрес съвпада с адреса на рутера
    - връща 1 ако този адрес фигурира в списъка от пътища
    - ако адресът не е в списъка и `hop_count > 1` то рутерът пита последователно всеки от съседите си, като им подава `hop_count - 1`
    - ако някой от съседите върне 1:
        - в списъка от пътища се добавя нов елемент с подадения адрес и индекса на съседа
        - методът връща 1 защото е намерен такъв път
    - ако никой съсед не върне 1 методът връща 0
- метод `void send_package(const Package& package)`, който опитва да изпрати подадени пакет
    - ако адресът, към който е изпратен пакетът, съответства на адреса на рутера, то пакетът е стигнал целта си и няма нужда от повече препращания
    - ако адресът, фигурира в списъка от пътища, то съответният брой изпратени пакети се увеличава с 1 и пакетът се препраща чрез `send_package` на съседния рутер
    - в противен случай се търси път с `query_route`, на който за първоначален `hop_count` се подава членът за брой скокове
    - ако не бъде намерен път за пакета той бива изхвърлен и не се праща никъде
    - методът да изписва съобщения в конзолата за всяко свое действие и резултат
- на всеки 10 изпратени пакета списъкът от пътища се сортира според броя пратени пакети в намаляващ ред
- при добавяне на познат път ако списъкът е пълен то новият елемент замества последния в списъка
- да се хвърлят изключения когато:
    - добавя се съсед с адрес `0.0.0.0` или `127.0.0.0`
    - изпраща се пакет без съдържание
    - изпраща се пакет с адрес `0.0.0.0` или `127.0.0.0`

3. Програмата да чете информация от файлове:

- `routers.txt` - списък от рутери. `name` е името на рутера, `address` е неговият адрес

```
<name1> <address1>
<name2> <address2>
...
<nameN> <addressN>
```

- `network.txt` - списък от връзките между рутери. `name` са имената на двойките свързани рутер. **Връзките са двупосочни**

```
name1 name2
name3 name4
...
nameN nameM
```

- `packages.txt` - списък от пакети, които да бъдат пратени. `source address` е адресът, от който се изпраща пакетът, `target address` е адресът, към който се изпраща, `content` е неговото съдържание. **Съдържанието е в кавички за да може да се тества и с празни стрингове**

```
<source address 1> <target address 1> "<content 1>"
<source address 2> <target address 2> "<content 2>"
...
<source address N> <target address N> "<content N>"
```