Задача 1. Да се реализира клас `NetworkDevice`, описващ мрежово устройство, което се идентифицира с етикет — символен низ (не непременно уникален) и капацитет: цяло число n, указващо максималния брой други устройства, с които може да бъде свързано (1 ≤ n ≤ 256). Класът да поддържа следните методи:

Конструктор с параметри етикета на устройството и капацитет

`bool attachTo (NetworkDevice* device)`, който свързва устройството с друго устройство. Операцията не е възможна, ако някое от устройствата не може да бъде свързано с повече устройства или ако двете устройства са вече свързани помежду си. Методът връща дали свързването е било успешно. Създадената връзка е двупосочна.

`unsigned attachedDevices () const`, който дава броя на текущо свързаните устройства към даденото устройство.

`NetworkDevice* getAttachedDevice (int i) const`, който връща i-тото поред устройство, към което даденото устройство е свързано, или `nullptr`, ако такова няма.

`bool isAttachedTo (NetworkDevice const* device) const`, който проверява дали две устройства са свързани

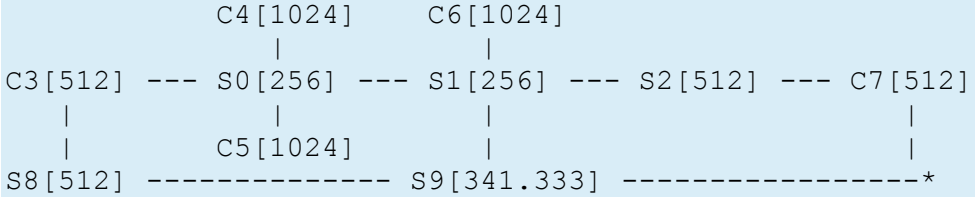
`double currentThroughput () const`, който връща пропускливостта на устройството. "Пропускливост" на дадено устройство d наричаме отношението на константата 1024 (отговаряща на скорост на пренос на данни 1Gbps) към броя на текущо свързаните устройства. Ако към устройството няма свързани други устройства, пропускливостта му е 0.

"Маршрут" между d_1 и d_k ще наричаме всеки път $d_1,...,d_k$ в графа от устройства. "Пропускливост" на маршрут наричаме минималната измежду пропускливостите на всички устройства в даден маршрут. Да се дефинира функция

```
std::vector<std::string> maxThroughputRoute (NetworkDevice const *d1,
                                             NetworkDevice const *d2);
```

която връща последователността от етикетите на устройствата в маршрут между d1 и d2 с максимална пропускливост.

Пример: На следната схема са изобразени имената на свързани устройства и връзките между тях, като в скоби са посочени пропускливостите им.



Маршрутът между C3 и C7 с максимална пропускливост се състои от устройствата с етикети "C3", "S8", "S9" и "C7".

Задача 2. Даден е HTML документ, отварящите и затварящите тагове в него трябва да са правилно вложени.

Приемаме, че имената на таговете се състоят единствено от малки латински букви, а текстовете не съдържат символа ‘<’. Всички trailing whitespaces се игнорират.

Разполагате с клас `HTMLIterator`, който обхожда “токените”, описващи отделните синтактични единици в HTML документа: тагове и низове. Токените се представят със следната структура: `struct Token { char type; string data; };`

Полето `type` описва типа на токена, и може да е един от следните символи:

'o' за отварящ таг, 'c' за затварящ таг, 't' за низ

Полето `data` описва съдържанието на токена – името на тага или съдържанието на текста

Класът `HTMLIterator` съдържа следните методи:

Конструктор `HTMLIterator(istream& is)`, който инициализира итератор с входен поток, съдържащ HTML документ

Операции * и `++`.

Оператор `bool`, който връща `false` <=> итераторът сочи END.

Пример: итераторът ще върне последователно следните токени: `{ 'o', "html"}, { 'o', "head"}, { 'o', "title"}, { 't', "Писмен изпит по СДП"}, { 'c', "title"}, { 'c', "head"},` ...

```
<html>
      <head>
            <title>Писмен изпит по СДП</title>
      </head>
      <body>
            <h1>Вариант 1</h1>
            <h2>Задача 1</h2>
            Текст на задача 1.
            <h1>Вариант 2</h1>
            TODO
      </body>
</html>
```

Напишете ф-я `vector<string> collect (HTMLIterator it, vector<string> tags);` която връща списък от всички низове в HTML документа, обхождан от итератора `it`, които са непосредствено вложени в последователност от тагове, описана чрез `tags`. Функцията трябва също да проверява дали таговете в HTML документа, прочетен от потока, са правилно вложени, и ако това не е вярно, да връща списък от единствен низ "ERROR".

Пример 1: За `tags` = "body", "h1", резултатът е списък от "Вариант 1", "Вариант 2".
Пример 2: За `tags` = "html", "body", резултатът е списък от "Текст на задача 1.", "TODO".
Пример 3: За `tags` = "html", "head", резултатът е празният списък.

Задача 3. Да се реализира клас `NetworkDevice`, описващ мрежово устройство, което се идентифицира с етикет — символен низ (не непременно уникален) и капацитет: цяло число n, указващо максималния брой други устройства, с които може да бъде свързано ($1 \leq n \leq 256$). Класът да поддържа следните методи:

Конструктор с параметри етикета на устройството и капацитет.

`bool attachTo (NetworkDevice* device)`, който свързва устройството с друго устройство. Операцията не е възможна, ако някое от устройствата не може да бъде свързано с повече устройства или ако двете устройства са вече свързани помежду си. Методът връща дали свързването е било успешно. Създадената връзка е двупосочна.

`unsigned attachedDevices () const`, който дава броя на текущо свързаните устройства към даденото устройство.

`NetworkDevice* getAttachedDevice (int i) const`, който връща i-тото поред устройство, към което даденото устройство е свързано, или `nullptr`, ако такова няма.

`bool isAttachedTo (NetworkDevice const* device) const`, който проверява дали две устройства са свързани

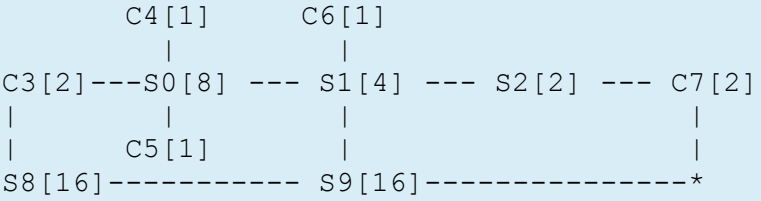
`double currentLoad () const`, който връща натовареността на устройството. "Натовареност" на дадено устройство d наричаме отношението на броя на устройствата, с които d текущо е свързано, към капацитета му.

"Маршрут" между d_1 и d_k ще наричаме всеки път $d_1,..,d_k$ в графа от устройства. "Натовареност" на маршрут наричаме максимума от натовареностите на всички устройства в даден маршрут, освен първото и последното. Ако маршрутът се състои само от две устройства, приемаме че натовареността му е 1. Да се дефинира функция

```
std::vector<std::string> minLoadRoute (NetworkDevice const *d1,
                                       NetworkDevice const *d2);
```

която връща последователността от етикетите на устройствата в маршрут между d1 и d2 с минимална натовареност.

Пример: На следната схема са изобразени имената на свързани устройства и връзките между тях, като в скоби са посочени капацитетите им.



Маршрутът между C3 и C7 с минимална натовареност се състои от устройствата с етикети "C3", "S8", "S9" и "C7".

Задача 4. Даден е HTML документ, отварящите и затварящите тагове в него трябва да са правилно вложени.

Приемаме, че имената на таговете се състоят единствено от малки латински букви, а текстовете не съдържат символа ‘<’. Всички trailing whitespaces се игнорират.

Разполагате с клас `HTMLIterator`, който обхожда “токените”, описващи отделните синтактични единици в HTML документа: тагове и низове. Токените се представят със следната структура: `struct Token { char type; string data; };`

Полето `type` описва типа на токена, и може да е един от следните символи:

'o' за отварящ таг, 'c' за затварящ таг, 't' за низ

Полето `data` описва съдържанието на токена – името на тага или съдържанието на текста

Класът `HTMLIterator` съдържа следните методи:

Конструктор `HTMLIterator(istream& is)`, който инициализира итератор с входен поток, съдържащ HTML документ

Операции * и ++.

Оператор `bool`, който връща `false` <=> итераторът сочи END.

Пример: итераторът ще върне последователно следните токени: `{ 'o', "html"}, { 'o', "head"}, { 'o', "title"}, { 't', "Писмен изпит по СДП"}, { 'c', "title"}, { 'c', "head"},` ...

```
<html>
      <head>
            <title>Писмен изпит по СДП</title>
      </head>
      <body>
            <h1>Вариант 1</h1>
            <h2>Задача 1</h2>
            Текст на задача 1.
            <h1>Вариант 2</h1>
            TODO
      </body>
</html>
```

“Път” наричаме последователност от вложени тагове, започваща от най-външния таг на документа, в която таговете са разделени с `'/'`. Низ е “достижим” по даден път **p**, ако последователността от тагове, в която той е вложен, започва с **p**.

Пример: “Писмен изпит по СДП” е достижим от пътищата `"html"`, `"html/head"` и `"html/head/title"`.

Напишете ф-я `string findAll(HTMLIterator it, vector<string> strings);`
която връща най-дългия път в документа, обхождан от итератора `it`, от който са достижими всички срещания на низове от вектора `strings`. Ако никой от низовете от

`strings` не се среща в документа, функцията да връща празния низ. Функцията трябва също да проверява дали таговете в HTML документа, прочетен от потока, са правилно вложени, и ако това не е вярно да връща единствен низ "ERROR".

Пример 1: За `strings` = "Вариант 1", "Задача 1" резултатът е "`html/body`"
Пример 2: За `strings` = "Вариант 1", "Вариант 2" резултатът е "`html/body/h1`"
Пример 3: За `strings` = "Писмен изпит по СДП", "TODO" резултатът е "`html`"