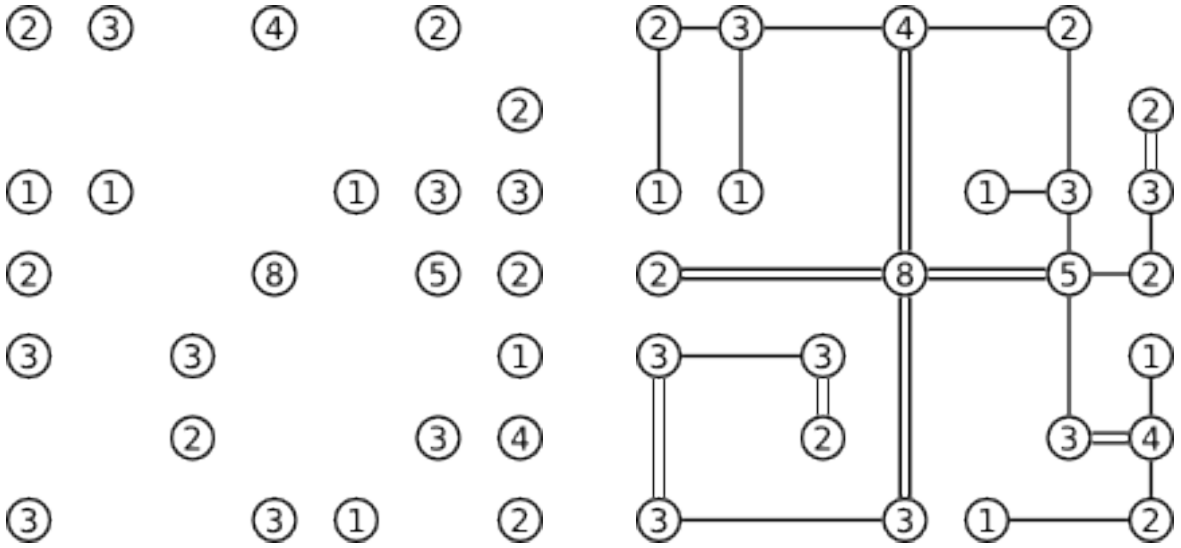
Task A2. ПЪЗЕЛ

⏳10 sec. 💾 256 MB

Алиса много обича да решава пъзели като судоку и други. Наскоро си намери нов такъв. Играе се на правоъгълна таблица. Някои клетки започват с числа от 1 до 8 включително; това са “островите”. Останалите клетки са празни. Никои два острова не се допират (т.е. никои два нямат общо страна). Целта е да свържете всички острови, като начертаете поредица от мостове между островите. Мостовете трябва да отговарят на определени критерии:

- Трябва да започват и завършват в различни острови, като образуват права линия между тях.
- Могат да са само хоризонтални или вертикални (т.е. не могат да са диагонални).
- Не трябва да пресичат други мостове или острови.
- Най-много два моста свързват двойка острови.
- Броят на мостовете, свързани с всеки остров, трябва да съвпада с числото на този остров.
- Мостовете трябва да свързват островите в една свързана група.

Имайте предвид, че решението може да не е единствено. Примерен екземпляр на този пъзел е показан отляво, а отдясно е едно от неговите решения (таблицата не е показана):



Алиса иска помощ с решаването на някои пъзели от този тип с различни размери. Помогнете ѝ, като напишете програма, която прави това. Ясно е, че тази задача може да няма бързо решение в най-лошия случай, така че е важно да се фокусираме върху “реалистични” случаи. За тази цел Ви е предоставен набор от тестове, който е еквивалентен на набора от тестове в системата за оценяване (генериран по същия начин, със същите параметри, само с различни начални сийдове).

Всеки тест ще се състои от един или повече подтестове – всеки от които е екземпляр на пъзела. Вашата програма ще бъде компилирана заедно с грейдър, който ще ѝ дава тези подтестове един по един и ще спре, когато програмата Ви върне неправилно решение, или когато програмата Ви реши да спре, или ако времето е изтекло, преди да е завършен подтестът. Вашият резултат за теста ще бъде пропорционален на броя успешно решени подтестове.

Подробности за реализацията

Първо грейдърът ще извика Вашата функция `init`, за да посочи броя на подтестовете и времевия лимит в секунди за теста (различните тестове може да имат различни времеви лимити):

```
void init(int numSubtests, double timeLimit);
```

След това за всеки подтест ще извика Вашата функция `solve` с екземпляр на пъзела, представен като вектор от вектори (вътрешните вектори представляват редове), където празните клетки са нули. Вашата функция трябва да модифицира тази структура, за да напише своето решение. Трябва да постави `-1` за единичен хоризонтален мост, `-3` за двоен хоризонтален мост, `-2` за единичен вертикален мост и `-4` за двоен вертикален мост. Трябва да върне `true`, ако иска да реши текущия подтест и да продължи, или да върне `false`, ако иска да спре (т.е. да не реши текущия подтест). Ето сигнатурата на функцията:

```
bool solve(std::vector<std::vector<int>>& puzzle);
```

И накрая, Вашите функции могат да викат следната функция, имплементирана в грейдъра, която връща изминалото време в секунди:

```
double timePassed();
```

Вашият код трябва да имплементира `init` и `solve`. Освен тях той може да има други помощни функции, променливи, структури и т.н. Кодът обаче не може да съдържа `main` функция и трябва да включва хедъра `puzzle.h`.

Локално тестване

Дават ви се файловете `Lgrader.cpp` и `puzzle.h`, които можете да компилирате заедно с Вашия код, за да го тествате. Даден Ви е и представителен набор от примерни тестове.

Ограничения

- $2 \leq \text{numRows}, \text{numColumns} \leq 47$
- $2 \leq \text{numIslands} \leq 200$

Тестове

Tests	numIslands	numRows, numColumns	numSubtests	timeLimit
1-3	≤ 15	≤ 7	$= 1$	2
4-6	≤ 15	≤ 7	$= 2$	2
7-8	≤ 100	≤ 31	$= 1$	2
9-10	≤ 100	≤ 31	$= 6$	2
11-12	≤ 100	≤ 31	$= 36$	2
13-14	≤ 200	≤ 47	$= 1$	5
15-16	≤ 200	≤ 47	$= 6$	5
17-18	≤ 200	≤ 47	$= 24$	8

Всички тестове се оценяват независимо и имат еднаква стойност.

Оценяване

Грейдърът многократно ще извиква `solve` с екземпляри от пъзела. Той ще спре да прави това и ще прекрати програмата, ако се случи някое от следните:

- Времевия лимит за теста е достигнат преди функцията Ви да върне стойност.
- Вашата функция връща невалидно решение.
- Вашата функция решава да спре (т.е. връща `false`).

Тогава ще получавате точките за всички правилно решени подтестове преди това, т.е. ако сте решили C подтеста, а има общо T подтеста, ще получите C/T от точките за теста.

Обърнете внимание, че трябва да избягвате да надвишите ограничението от 10 секунди на системата за оценяване. Можете да използвате `timePassed()` и спиране чрез връщане на `false` за това. Ако програмата Ви го надвиши, системата ще я убие и е получите 0 точки.

Примерна комуникация

Първо се извиква `init(int 1, double 2);`. След това:

```
solve({
    { 0,  0,  2,  0,  3,  0,  3},
    { 4,  0,  0,  0,  0,  2,  0},
    { 0,  0,  0,  0,  0,  0,  0},
    { 6,  0,  0,  0,  0,  2,  0},
    { 0,  0,  0,  0,  0,  0,  0},
    { 0,  0,  0,  0,  0,  0,  0},
    { 4,  0,  0,  0,  0,  0,  4}
});
```

Тази функция връща `true` и променя дадения вектор, както следва:

```
{
    { 0,  0,  2, -3,  3, -1,  3},
    { 4, -3, -3, -3, -3,  2, -4},
    {-4,  0,  0,  0,  0,  0, -4},
    { 6, -3, -3, -3, -3,  2, -4},
    {-4,  0,  0,  0,  0,  0, -4},
    {-4,  0,  0,  0,  0,  0, -4},
    { 4, -3, -3, -3, -3, -3,  4}
}
```

Това ще осигури пълните точки за теста, ако е в определения лимит на време.