Zadanie 1 - Napisz program, który na podstawie wyników wyborów w poszczególnych okręgach odtworzy podział mandatów zgodnie z ordynacją wyborczą do Sejmu RP.

Program powinien ze standardowego wejścia czytać dane takie, jak w załączonych plikach. W pierwszym wierszu znajdują się "nagłówki kolumn", czyli kolejno słowa "okreg", "mandatow" (które właściwie można zignorować, są tu dla czytelności) oraz (skrótowe) nazwy komitetów wyborczych. Każdy z pozostałych wierszy zawiera informacje o wynikach wyborów w danym okręgu wyborczym i zawiera kolejno nazwę tego okręgu (w przygotowanych plikach są one liczbowe, ale nie zakładaj, że tak będzie zawsze, zresztą w tym zadaniu je też można zignorować), liczbę mandatów sejmowych do podziału w tym okręgu, oraz liczby głosów oddanych w tym okręgu na kolejne komitety wymienione w nagłówku. Brak głosów może być podany zarówno jako zero, jak i pusty napis; inne pola nie mogą być puste. Sąsiednie pola oddzielone są średnikami, a wejście należy czytać do końca (czyli najlepiej testować program przekierowując na nie któryś z plików, tj. wykonując ./a.out <wyniki2019.csv).

W pierwszym kroku procedury podziału mandatów należy dla każdego komitetu określić, czy w skali całego kraju przekroczył próg wyborczy 5% wszystkich oddanych głosów – pozostałe komitety są ignorowane w dalszych rozważaniach. Następnie w każdym okręgu rozważa się zbiór tzw. ilorazów – wyników dzielenia liczb głosów oddanych na poszczególne komitety przez kolejne liczby naturalne (od 1). n mandatów przypadających na dany okręg dostają te komitety, których ilorazy są wśród n największych. Ewentualne remisy rozstrzygane są na korzyść komitetu, który dostał w danym okręgu więcej głosów (załóż, że dalszych remisów nie będzie). Tę część zrealizuj sortując przy użyciu qsort z stdlib.h (z odpowiednim komparatorem) tablicę rekordów zawierających wszystkie potrzebne informacje (np. iloraz i całkowitą liczbę głosów) oraz indeks komitetu, któremu odpowiada ten iloraz. Liczby mandatów uzyskane przez poszczególne komitety we wszystkich okręgach program ma wysumować i czytelnie wydrukować (pomijając komitety, które nie uzyskały mandatów).

Program powinien kończyć wykonanie drukując informację o błędzie, kiedy dane okażą się niepoprawne, np. gdy jest za mało kolumn (musi być co najmniej jeden komitet), gdy w dalszych wierszach jest inna liczba kolumn (tj. średników) niż w wierszu nagłówkowym, lub gdy nazwy komitetów powtarzają się.

Dodatkowo zaimplementuj obsługę następujących opcji wywołania:

--minority, po której następuje nazwa komitetu – oznacza to, że dany komitet reprezentuje mniejszość etniczną i nie obowiązuje go próg wyborczy (przykładowo, żeby odtworzyć faktyczne rezultaty wyborów w 2019 roku, będzie należało użyć opcji --minority MN);
--alliance, po której następuje ciąg nazw komitetów oddzielonych średnikami, i która ma dwa efekty:

wymienione komitety traktowane są w obliczeniach (oraz końcowych wynikach) jako jeden, a liczby otrzymanych przez nie głosów sumują się;
komitet taki obowiązuje wyższy próg wyborczy 8% (i dlatego może być sensowne użycie tej opcji dla pojedynczego komitetu, np. odtworzenie faktycznych rezultatów wyborów w 2015 roku wymagałoby użycia opcji --alliance ZL);
--sainte-lague (bezargumentowa), z którą ilorazy otrzymuje się dzieląc przez kolejne liczby nieparzyste (a nie wszystkie naturalne).
Również dla dowolnych bezsensownych (zestawów) opcji wywołania program powinien kończyć wykonanie drukując informacje o błędzie, np. gdy opcja --sainte-lague powtarza się (pozostałe mogą występować wielokrotnie), lub w argumentach pozostałych opcji pojawiają się komitety powtórzone bądź nieistniejące.

Przemyśl organizację kodu, w tym podział na funkcje i ew. wydzielenie modułu/-ów.

Ze względu na specjalne znaczenie średnika w terminalu argument opcji --alliance będzie należało np. podawać w cudzysłowie (który będzie przetwarzany przez terminal i "zniknie" przed pojawieniem się napisu w programie), np. --alliance "KWIN;BS".

Zadanie 2 - W jaskini znajduje się n^2 żarówek, uporządkowanych w siatkę n na n.

Każda żarówka powoli gromadzi energię i na moment rozbłyska, gdy osiągnie odpowiedni stan.

Na przykład:
5483143223
2745854711
5264556173
6141336146
6357385478
4167524645
2176841721
6882881134
4846848554
5283751526

Poziom energii każdej żarówki to wartość od 0 do 9. Żarówka w lewym górnym rogu ma poziom energii 5, ta w prawym dolnym rogu ma poziom energii 6, i tak dalej. Możesz modelować poziomy energii i rozbłyski światła w krokach.

Podczas pojedynczego kroku zachodzi następująca sekwencja: 1. Najpierw poziom energii każdej żarówki zwiększa się o 1. 2. Następnie każda żarówka z poziomem energii większym niż 9 rozbłyska.

To zwiększa poziom energii wszystkich sąsiednich żarówek o 1 (włączając żarówki położone po przekątnej). Jeśli to powoduje, że żarówka osiąga poziom energii większy niż 9, ona również rozbłyska. Ten proces trwa tak długo, jak nowe żarówki przekraczają poziom energii 9 (żarówka może rozbłysnąć co najwyżej raz na krok). Na koniec, każda żarówka która rozbłysła podczas tego kroku ma swój poziom energii ustawiony na 0, ponieważ zużyła na to całą swoją energię.

Sąsiednie rozbłyski mogą spowodować, że żarówka rozbłyśnie w danym kroku, nawet jeśli zaczyna ten krok z bardzo małą ilością energii. Rozważ środkową żarówkę z energią 1 w tej sytuacji:

Stan początkowy:
11111
19991
19191
19991
11111

Po kroku 1:
34543
40004
50005
40004
34543

Po kroku 2:
45654
51115
61116
51115
45654

Twoim zadaniem jest napisać program, który obliczy ilość rozbłysków po m krokach.

Dane wejściowe:
n m
$c_{0,0}$ $c_{0,1}$ ... $c_{0,□-1}$
$c_{1,0}$ $c_{1,1}$ ... $c_{1,□-1}$
...
$c_{□-1,0}$ $c_{□-1,1}$ ... $c_{□-1,□-1}$

ograniczenia na dane:
n<50
m<20000000

n*m<50000000

Przykład 1:
5 3
11111
19991
19191
19991
11111

Odpowiedź:
9

Przykład 2:
10 10
5483143223
2745854711
5264556173
6141336146
6357385478
4167524645
2176841721
6882881134
4846848554
5283751526

Odpowiedź:
204

Przykład 3:
10 100
5483143223
2745854711
5264556173
6141336146
6357385478
4167524645
2176841721
6882881134
4846848554
5283751526

Odpowiedź:
1656