Zadanie 1 (10 pkt. na pracowni, później 5 pkt.). Napisz program, który wczyta ze standardowego wejścia jeden wiersz, następnie usunie z niego wszystkie znaki niebędące cyframi, po czym odpowie na pytanie, czy powstała liczba jest kwadratem liczby naturalnej.

Przykładowo, napisy "2dwadziescia5piec" i "!2#$56&*()" po usunięciu znaków innych niż cyfry tworzą kolejno liczby 25 oraz 256, które są kwadratami liczb naturalnych. Z kolei napisy "gsgs1fdsfs7dadsa" oraz "abc" po usunięciu znaków innych niż cyfry nie tworzą liczb, które są kwadratami liczb naturalnych.

Możesz założyć, że wiersz czytany z wejścia składa się z nie więcej niż 100 znaków, spośród których najwyżej 9 to cyfry.

Aby Twój kod był bardziej czytelny, we wszystkich miejscach gdzie jest to stosowne, wydziel fragmenty kodu do osobnych funkcji. Używaj też funkcji bibliotecznych, np. tych wspomnianych na ostatnim wykładzie.

Zadanie 2 (10 pkt.). Napisz program, który odczyta z argumentów wywołania liczbę *n*, a po niej *n* słów składających się tylko z (małych i dużych) liter alfabetu angielskiego. Pamiętaj o sprawdzeniu, czy liczba argumentów podanych na wejściu jest spójna z wartością *n*.

Program powinien policzyć, ile razy na wejściu wystąpiło każde słowo, **ignorując wielkość liter**, po czym wypisać wyniki na standardowe wyjście. Słowa te powinny być wypisane małymi literami, w dowolnej kolejności.

Następnie, Twój program powinien policzyć identyczną statystykę, ale biorąc pod uwagę **tylko wielkość liter**. W tym przypadku, przykładowo słowa AbCDe, EfGHz oraz AaAAa są sobie równe. Wypisując policzone statystyki użyj słów składających się tylko z liter a oraz A, a wyniki również możesz wypisać w dowolnej kolejności.

Przykład: dla wywołania funkcji ./a.out 5 aSdFg KUBEK ASDFG aSdFg inne odpowiedzią powinno być:

```
Statystyki ignorujace wielkosc liter:
asdfg 3
kubek 1
inne 1
Statystyki wielkosci liter:
aAaAa 2
AAAAA 2
aaaa 1
```

Za zrobienie dowolnej połowy zadania (tj. policzenie tylko jednego zestawu statystyk) można uzyskać połowę punktów.

Uwaga: program może nadpisywać słowa z tablicy argv (np. zamieniając wszystkie litery na małe), choć należy robić to ostrożnie, w szczególności nie bardzo da się te słowa "wydłużyć". To może jednak nie dawać wielkiej wygody w pełnej wersji zadania, skoro trzeba policzyć dwie rzeczy, przy czym dla drugiej istotne jest dokładnie to, co pomijane przy pierwszej. Być może warto będzie napisać jakieś własne funkcje.

Zadanie 3 (10 pkt.).

Na dnie kadłuba statku znajduje się zbiornik balastowy o szerokości 1 jednostki (o tym wymiarze nie będziemy więcej wspominać – interesuje nas tylko "płaski" widok z boku), pewnej długości i nierównym dnie (tj. zmiennej głębokości). W zbiorniku układane są obciążniki w kształcie sześcianu jednostkowego tak, że najpierw obciążnik opuszczany jest z góry przy lewym brzegu zbiornika, a potem:

1. jeśli przesunięcie go po równej powierzchni (innych obciążników bądź dna) maksymalnie w prawo zakończyłoby się osiągnięciem powierzchni **dna**, za którą jest obniżenie, to obciążnik jest opuszczany przy lewym brzegu tego obniżenia, a proces jest kontynuowany z tymi samymi regułami;
2. jeśli przesunięcie go jw. zakończyłoby się osiągnięciem powierzchni **innego obciążnika**, za którą jest obniżenie (w takiej sytuacji możliwe jest tylko obniżenie o 1), to obciążnik jest opuszczany przy lewym brzegu tego obniżenia, i tam pozostaje;
3. jeśli przesunięcie go jw. zakończyłoby się uderzeniem w ścianę (tj. wyższy fragment dna), to obciążnik pozostaje w obecnym miejscu (tj. nie jest przesuwany).

(Reguła 2. jest wydzielona z 1. tylko dla przejrzystości: w przypadku w niej opisanym można byłoby zastosować regułę 1. pozbawioną rozróżnienia rodzajów powierzchni przed obniżeniem, po czym musiałaby nastąpić sytuacja z reguły 3. Inny alternatywny opis: każdy obciążnik przesuwany jest po równym bądź opadającym dnie maksymalnie w prawo, a potem maksymalnie w lewo, i w tym miejscu zostaje.)

Przykładowo, jeśli zbiornik ma następujący przekrój (# – dno/ściany, . – pusta przestrzeń, * – obciążniki):

```
#......#
#......#
#...#..#
#####..#
########
```

to jego stan po ułożeniu 1, 3, 4, 5, 6, 9 i 16 obciążników (w ich docelowych położeniach – sam proces opisany dla pojedynczego obciążnika nie jest symulowany!) wyglądałby odpowiednio tak:

```
#......#
#......#
#*..#..#
#####..#
########

#......#
#......#
#***#..#
#####..#
########

#......#
#......#
#***#..#
#####*.#
########

#......#
#......#
#***#..#
#####**#
########

#......#
#......#
#***#*.#
#####**#
########

#......#
#**....#
#***#**#
#####**#
```

```
########

#***...#
#******#
#***#**#
#####**#
########
```

Ostatni przykład jest poprawny, mimo że dwa ostatnie obciążniki, zanim zostaną umieszczone na miejscu, przejściowo znajdują się "poza rysunkiem".

Zadanie

Program ma wczytać diagram przekroju pustego zbiornika i liczbę dostępnych obciążników, oraz wydrukować diagram przekroju zbiornika po umieszczeniu w nim tylu obciążników, ile się da.

Wejście

W pierwszym wierszu znajdują się trzy liczby naturalne, kolejno długość zbiornika *m*, jego głębokość *n*, i liczba odważników; 3 ≤ *m*, *n* ≤ 1000. W kolejnych *n* wierszach znajduje się po *m* znaków przedstawiających przekrój zbiornika: kratka # oznacza dno/ściany, a kropka . – pustą przestrzeń. Skrajne kolumny i ostatni wiersz zawierają same kratki; pierwszy wiersz, poza kratkami na skrajach, zawiera same kropki. W każdej kolumnie poniżej najwcześniejszej kratki są już same kratki. "Lewy brzeg zbiornika", przy którym według opisu początkowo pojawiają się obciążniki to najwcześniejsza kolumna zawierająca jakieś kropki, tj. ta o indeksie 1.

W testach 6-10 rozmiary diagramu są bliskie maksymalnym, ale w testach 6-7 liczba odważników jest taka, że wystarczy zastosować bardzo prostą heurystykę (tj. rozważyć specjalny przypadek, prosto obliczyć i wydrukować wynik, i przedwcześnie zakończyć działanie programu). W testach 1-5 diagram jest znacznie mniejszy.

Wyjście

Wyjście ma składać się z *n* wierszy po *m* znaków każdy i analogicznie przekrój zbiornika po umieszczeniu w nim obciążników (oznaczonych gwiazdkami *) zgodnie z opisem.

Przykłady

Przykład A

Wejście

```
5 5 6
#...#
#...#
##..#
##..#
#####
```

Wyjście

```
#...#
#**.#
##**#
##**#
#####
```

Przykład B

Wejście

```
8 5 6
#......#
#......#
#...#..#
#####..#
########
```

Wyjście

```
#......#
#......#
#***#*.#
#####**#
########
```

Przykład C

Wejście

```
4 3 6
#..#
#..#
####
```

Wyjście

```
#**#
#**#
####
```

Uwagi

Wczytanie znaków diagramu najprościej zrealizować, umieszczając w (być może zagnieżdżonej) pętli instrukcję scanf(" %c", &znak); – spacja jest tu istotna, bo będzie "zjadać" ewentualne białe znaki, czyli w tym przypadku końce wiersza, które, jak zwykle, są w przykładach głównie dla czytelności. Oczywiście w powyższym przykładzie zamiast znak można napisać tablica[indeks] itp., ale nie można zapomnieć o kaczorze &.