Задача A. Художник

Входной файл:	`painter.in`
Выходной файл:	`painter.out`
Ограничение по времени:	4 секунды
Ограничение памяти:	64 мегабайта

Не успев дорисовать свой гениальный футуристический шедевр, М. Калевич увлекся рисованием одномерных черно-белых картин. Он пытается найти оптимальное местоположение и количество черных участков картины. Для этого он проводит на прямой белые и черные отрезки, и после каждой из таких операций хочет знать количество черных отрезков на получившейся картине и их суммарную длину.

Изначально прямая — белая. Ваша задача — написать программу, которая после каждой такой операции выводит в выходной файл интересующие художника данные.

Описание входного файла

В первой строке входного файла содержится общее количество нарисованных отрезков ($1 \le N \le 100\,000$). В последующих N строках содержится описание операций. Каждая операция описывается строкой вида $c\ x\ l$, где c — цвет отрезка (`W` для белых отрезков, `B` для черных), а сам отрезок имеет вид $[x; x + l]$, причем координаты обоих концов — целые числа, не превосходящие по модулю 500 000. Длина задается положительным целым числом.

Описание выходного файла

После выполнения каждой из операций необходимо вывести в выходной файл на отдельной строке количество черных отрезков на картине и их суммарную длину, разделенные одним пробелом.

Пример

painter.in	painter.out
7 W 2 3 B 2 2 B 4 2 B 3 2 B 7 2 W 3 1 W 0 10	0 0 1 2 1 4 1 4 2 6 3 5 0 0

Задача B. Восьмиугольники

Входной файл:	`octagons.in`
Выходной файл:	`octagons.out`
Ограничение по времени:	2 секунды
Ограничение памяти:	16 мегабайт

На плоскости расположены n правильных восьмиугольников. Ваша задача — сосчитать число целых точек (то есть, точек, у которых обе координаты целые), покрытых как минимум k из этих многоугольников.

Описание входного файла

В первой строке дано целое число $1 \le n \le 500$. Последующие n строк описывают восьмиугольники. Каждый восьмиугольник задан координатами описаной окружности (x_c, y_c) и ее радиусом r_c. Восьмиугольники повернуты таким образом, что содержат только горизонтальные, вертикальные и диагональные отрезки. Координаты и радиусы — целые числа, не превосходящие 2000 по абсолютной величине.

Файл завершается строкой, содержащей число $1 \le k \le 500$.

Описание выходного файла

Выведите число целых точек, покрытых одновременно как минимум k восьмиугольниками. Точка считается покрытой восьмиугольником, если она лежит внутри или на стороне, либо является вершиной.

Пример

octagons.in	octagons.out
3 0 0 1 0 0 1 -1 0 1 2	1

Задача C. Крепкий орешек против мирового зла

Входной файл:	*нет*
Выходной файл:	*нет*
Ограничение по времени:	2 секунды
Ограничение памяти:	64 мегабайта

Человечество снова в опасности. Злой гений профессор Мариарти хочет взорвать планету с помощью изобретенной им кислородной бомбы. На борьбу со злом отправляется народный герой Уилл Брюсов, более известный как Крепкий орешек. Ночью под покровом темноты Крепкий орешек проник на территорию секретных лабораторий профессора. Он выяснил, что детонатор к бомбе хранится в гигантском холодильнике, где-то в тайнике, расположенном под землей. Холодильник представляет собой прямоугольное помещение разделенное на $N \times M$ квадратных секторов. По необъяснимым причинам холодильник обладает очень странной топологией. Каждый сектор имеет ровно четыре соседних сектора. Северным соседом для любого сектора, расположенного в самом северном ряду, является соответствующий ему сектор в самом южном ряду. Аналогичные условия верны для всех секторов, расположенных на границе холодильника. Крепкий орешек не задумывается над природой столь странного явления — его задача спасти человечество. Уилл попал в холодильник с помощью вертолета, и так как вокруг сплошная темнота, он не знает своей начальной позиции. За один ход Крепкий орешек может либо переместиться на одну из четырех соседних клеток, либо попытаться найти тайник. Если тайник в самом деле находится под землей в пределах сектора — Уилл его обязательно найдет. Хотя Уилл и супергерой, он теряет одну единицу своей жизненной силы при переходе на соседнюю клетку. Попытка поиска тайника отнимает у него пять единиц жизненной силы. Кроме того, некоторые сектора покрыты специальным составом (Уилл такие сектора называет «*скользкие места*»). При попадании на них Крепкий орешек падает и сильно ударяется головой: это отнимает у него десять единиц жизненной энергии. К счастью, в распоряжении Уилла есть карта холодильника. Таким образом, про каждый сектор известно, покрыт ли он специальным составом.

Вам предстоит испытать на себе все трудности, связанные со спасением человечества. Напишите программу, которая будет будет взаимодействовать с модулем. Программа должна принимать решения о том, как исследовать лабиринт. Ваша конечная цель — найти тайник. Программа не должна пытаться читать либо записывать информацию в какие-либо файлы на диске.

Гарантируется, что для любого предложенного вашей программе теста $N, M \leq 25$. Ваше решение будет оцениваться исходя из количества единиц жизненной энергии, потерянной в процессе поиска тайника. Конечно, Вы должны минимизировать это число.

Описание взаимодействия с модулем

Вам доступны следующие процедуры и функции:

`Init` — инициализация, возвращает размер лабиринта и его карту.

`DoMove` — с помощью вызова этой функции вы перемещаетесь на соседний сектор. В качестве результата эта функция возвращает `true` (для пишущих на языке C или C++ отличное от `0` значение), если клетка на которую было совершено перемещение является «*скользким местом*». Входным параметром для функции является целое число от `0` до `3`, обозначающее направление перемещения (`0` — на север, `1` — на восток, `2` — на юг, `3` — на запад). Ваша программа не должна делать более 6250 вызовов функции `DoMove`.

`Dig` — с помощью вызова этой функции вы пытаетесь найти тайник под землей в пределах текущего сектора. Функция возвращает `false` (или `0` для пишущих на языке C/C++) в случае неудачной попытки и заканчивает выполнение программы в противном случае (в этом случае ваша миссия выполнена). Ваша программа не должна делать более 625 вызовов функции `Dig`.

Ваша программа должна вызвать процедуру `Init` для получения параметров задачи, после этого использовать функции `DoMove`, `Dig` для перемещения по холодильнику и поиска тайника.

На время тура вам доступен демонстрационный модуль. Этот модуль получает данные для задачи из файла `diehard.in`. В первой строке этого файла должно быть записано два натуральных числа N, M — количество горизонтальных и вертикальных рядов секторов в холодильнике. Далее в N строках должна содержаться карта холодильника. Символ `'*'` обозначает, что данный сектор является «*скользким местом*», `'T'` — сектор, под которым расположен тайник, `'S'` — ваша начальная позиция, `'.'` — просто пустой сектор. Заметим, что в той карте, которая возвращается процедурой `Init`, символ `'S'` в карте заменяется на `'.'`, так как начальная позиция супергерою неизвестна. Известно, что сектор на который приземлился супергерой, как и сектор под которым расположен тайник, не является «*скользким местом*».

Инструкция для пишущих на языке Pascal:

Вам доступны два файла: "`diehard.ppw`" и "`diehard.ow`". Данные файлы представляют собой реализацию модуля, интерфейсная часть приведена ниже:

```
const
  NMAX = 25;

  DIR_UP = 0;
  DIR_RIGHT = 1;
  DIR_DOWN = 2;
  DIR_LEFT = 3;
  DIG_CHAR = 'v';
  MAP_ICE = '*';
  MAP_EMPTY = '.';
  MAP_T = 'T';
  DMG_MOVE = 1;
  DMG_FALL = 10;
  DMG_DIG = 5;

type
  TMap = array [0..NMAX, 0..NMAX] of char;

procedure Init(var n, m: longint; var map: TMap);
function DoMove(dir: longint): boolean;
function Dig: boolean;
```

Инструкция для пишущих на языках C и C++:

Вам доступны два файла: "`diehard.h`" и "`diehard.o`". Данные файлы представляют собой реализацию модуля, интерфейсная часть которого приведена ниже:

```
#define NMAX 25

#define DIR_UP 0
#define DIR_RIGHT 1
#define DIR_DOWN 2
#define DIR_LEFT 3
#define DIG_CHAR 'v'
#define MAP_ICE '*'
#define MAP_EMPTY '.'
#define MAP_T 'T'
#define DMG_MOVE 1
#define DMG_FALL 10
#define DMG_DIG 5

void Init(int* n, int* m, char** map);
int DoMove(int dir);
int Dig();
```

Пример

Пример файла `diehard.in` для демонстрационного модуля.

```
3 3
...
S*T
...
```