B. Через тернии к клиенту

|  | Все языки | Clang 16.0.0 C++20 | GNU GCC 12.2 C++20 |
| --- | --- | --- | --- |
| Ограничение времени | 5 секунд | 1 секунда | 1 секунда |
| Ограничение памяти | 512Mb | 512Mb | 512Mb |
| Ввод | стандартный ввод или input.txt | | |
| Вывод | стандартный вывод или output.txt | | |

Известная компания Тындекс идёт в ногу со временем — с началом активных космических перелётов в компании открылся сервис Тындекс.Ракетакси: пользователю необходимо лишь указать координаты начала и конца перелёта, после чего за ним вылетит персональная ракета.

По сути любой заказ можно описать в виде событий четырёх типов:

1. **A**(accepted) - заказ принят в работу (ракета вылетела за клиентом);
2. **B**(boarding) - клиент сел в ракету;
3. **S**(success) - заказ успешно завершён (клиент вышел на планете назначения);
4. **C**(cancelled) - заказ отменён.

Все происходящие с ракетами события отправляются на сервера, где сразу логируются. Вот только из-за проблем со связью (метеоритные потоки, вспышки на звездах и т.д.) отправка событий иногда затягивается, из-за чего записи в получившемся логе могут идти не по порядку.

Гарантируется, что все описанные в логе события задают один из следующих сценариев:

1. **A**- **B**- **S**
2. **A**- **B**- **C**
3. **A**- **C**

Вам, как главному аналитику (и по совместительству главному программисту) ракетопарка, необходимо исследовать лог за прошедший год и определить для каждой ракеты суммарное время движения (в минутах) в течение заказов.

В каждый момент времени ракета выполняет только один заказ. Будем считать, что каждая ракета в каждый момент времени:

* или стоит на месте в ожидании заказа,
* или перемещается по космосу, выполняя заказ.

Движение начинается после принятия заказа и завершается после отмены или завершения заказа. За одну минуту не может произойти несколько событий, связанных с одной и той же ракетой.

Формат ввода

В первой строке дано целое число N(2≤N≤200000)  — количество записей в логе.

Следующие N строк содержат записи в логе в формате dayhourminuteidstatus:

* day(1≤d≤365)  — номер дня (сквозная нумерация с начала календарного года);
* hour(0≤h<24)  — часы;
* minute(0≤m<60)  — минуты;
* id(0≤id≤109)  — уникальный идентификатор ракеты;
* status∈{A,B,S,C}  — буква, обозначающая тип события.

Формат вывода

В единственной строке выведите через пробел суммарное время движения на заказах для каждой упомянутой в логе ракеты. Данные необходимо выводить в порядке возрастания идентификаторов ракет.

Пример

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| 8  50 7 25 3632 A  14 23 52 212372 S  15 0 5 3632 C  14 21 30 212372 A  50 7 26 3632 C  14 21 30 3632 A  14 21 40 212372 B  14 23 52 3632 B | 156 142 |

Примечания

**Ракета №3632**

1. в 14-й день года в 21:30 получила заказ (шестая запись в логе);
2. забрала пассажира в 23:52 того же дня (восьмая запись в логе);
3. после чего заказ был отменён в 15-й день года в 00:05 (третья запись в логе);
4. в 50-й день года в 7:25 получила заказ (первая запись в логе);
5. заказ был отменён уже через минуту (четвёртая запись в логе).

Таким образом ракета №3632 провела в движении с 14-го дня 21:30 до 15-го дня 00:05 и с 50-го дня 7:25 до 50-го дня 7:26 — всего 156 минут.

**Ракета №212372**

1. в 14-й день года в 21:30 получила заказ (третья запись в логе);
2. через 10 минут забрала пассажира (седьмая запись в логе);
3. в 23:52 прибыла на место назначения (вторая запись в логе).

Всего ракета №212372 провела в движении с 14-го дня 21:30 до 14-го дня 23:52 — 142 минуты.