Campeonato de Futebol

Futebol é o esporte mais popular do mundo e, como todos sabem, existem milhares de times que participam em campeonatos ao redor do mundo inteiro.

Com tantos times, é dura a tarefa manter registro dos jogos e dos placares de todos os campeonatos. Portanto, sua tarefa é a seguinte: escreva um programa que recebe o nome de um campeonato, os nomes dos times e uma descrição de todos os jogos. Imprima a tabela do campeonato.

Cada jogo pode terminar em vitória de um dos times ou empate. Não há diferença entre jogos em casa ou como visitante. O time vencedor leva 3 pontos e, em caso de empate, ambos os times levam 1 ponto.

O placar geral do campeonato deve ser ordenado da seguinte forma (**use MergeSort, QuickSort ou HeapSort**):

- O primeiro critério é o número de pontos. Um time que fez mais pontos, obviamente, estará melhor classificado que os times com menor número de pontos;
- O segundo critério é o número de vitórias. Se dois times tiverem o mesmo número de pontos, então o time que tiver mais vitórias estará melhor classificado;
- O terceiro critério é a diferença de gols. Se dois times empatarem nos critérios anteriores, então o time que tiver a maior diferença entre gols feitos e gols recebidos será melhor classificado;
- O quarto critério é o número de gols marcados. Se dois times empatam em todos os critérios anteriores, então o time que marcou mais gols será melhor classificado;
- O quinto critério é o número de partidas jogadas. Se dois times empatam em todos os critérios anteriores, então o time que realizou menos jogos até o momento será melhor classificado;
- O último critério é o nome do time. Se dois times estão iguais em todos os critérios, então o time com nome de menor ordem lexicográfica estará melhor classificado.

Entrada

A entrada será uma descrição completa de vários campeonatos. Cada campeonato é um caso de teste.

O primeiro valor na entrada será um inteiro N indicando o número de campeonatos.

Cada campeonato é uma descrição dos times e de todos os jogos realizados. A descrição do campeonato começa com o nome do campeonato, que é uma *string* sem espaços que você pode ler com `input()`.

Em seguida, aparecerá um inteiro T$\square$ indicando o número de times ($1<T<=30$), seguido de T$\square$ palavras com os nomes dos times. Os nomes dos times podem conter letras maiúsculas e minúsculas, números, e o caracter barra-baixa (_).
Após o nome do último time, haverá um inteiro G$\square$ indicando o número de jogos realizados até agora.
Os G$\square$ jogos serão descritos da seguinte forma: o nome de um time, o separador `:`, o número de gols marcados por esse time, o separador `#`, o nome do segundo time, o separador `:` e o número de gols marcados pelo segundo time. Por exemplo, a descrição `Bananeiras:3#Sao_Pedro:2` indica que a equipe Bananeiras venceu o time do São Pedro por 3 a 2. Todos os gols são números não negativos menores que 20. Você pode presumir que todos os nomes que aparecem nos jogos são times que foram apresentados anteriormente. Nenhum time irá jogar contra si mesmo.

Saída

Para cada campeonato, imprima primeiramente o nome do campeonato em uma linha. As próximas T$\square$ linhas devem descrever os T$\square$ times do campeonato, ordenados de acordo com as regras apresentadas anteriormente.
Para cada time, imprima uma linha com o seguinte formato:

```
[a] - Nome_do_time: [b] pontos, [c] jogos ([d]-[e]-[f]), d.g. [g] ([h]-[i])
```

Substituindo:

- [a] pela posição do time no placar
- [b] pelo total de pontos que o time conseguiu
- [c] pelo número de jogos que o time jogou
- [d] pelo número de vitórias
- [e] pelo número de empates
- [f] pelo número de derrotas
- [g] pela diferença de gols (marcados - sofridos)
- [h] pelo número de gols marcados
- [i] pelo número de gols sofridos

Imprima uma linha em branco após cada campeonato.

Restrição

Você deve utilizar um dos algoritmos de ordenação quasi-lineares vistos em aula: MergeSort, QuickSort ou HeapSort.

Dicas

- Use dicionários ou crie uma classe para guardar os dados de cada time.

- Faça uma função para verificar se um time está melhor classificado que outro. Use essa função para auxiliar o desenvolvimento da sua função de ordenação.

- O Python permite comparar diretamente duas strings em ordem lexicográfica, mas diferencia maiúsculas de minúsculas. Por exemplo, a expressão `"alfa" < "beta"` é verdadeira, mas a expressão `"alfa" < "BETA"` é falsa. Você pode usar o método `str.lower` para converter os caracteres em letras minúsculas antes de comparar.

- A descrição dos jogos é mais simples do que parece. Use o método `str.split` para separar a descrição dos dois times e depois use o mesmo método para separar o nome da pontuação de cada time.

- Modularize bem seu código! Por exemplo, você pode começar fazendo uma função para processar a descrição de um único jogo. Essa função pode ler uma linha com `input` e, usando `str.split`, retornar quatro valores. Essa função pode ser usada por uma outra função chamada `ler_jogos`, que poderá ser chamada por uma função para processar o campeonato etc.

- Para processar os jogos e calcular os dados de cada time, você precisará usar uma estrutura que permita fazer busca pelos nomes dos times. Como são poucos times, você pode usar uma lista comum e fazer busca linear. Ou, se preferir, uma ideia mais interessante é usar um `dict` no qual a chave é o nome do time e o valor contém os dados (total de jogos, gols marcados e sofridos, pontos etc.).

- Depois que você processar todos os jogos, será necessário ordenar os times pelos critérios do exercício. Se você criou uma classe para os times, você pode implementar agora o método `__lt__`, que deverá retornar `True` se o objeto for "menor" que outro. Considere que um objeto é "menor" se ele deve aparecer *antes* no placar, ou seja, se ele foi *melhor* no campeonato.

- Você pode usar o método `list.sort` para testar a sua solução antes de imeplementar o seu algoritmo de ordenação. Assim você pode verificar primeiro a leitura, o processamento dos jogos e a impressão antes de testar seu algoritmo de ordenação.

- Lembre-se: o melhor jeito de resolver um problema complexo é abordando uma parte por vez.

Exemplos de Entrada e Saída

Entrada

```
2
Torneio-dos-Programadores
3
Dijkstra
Wirth
Knuth
4
Dijkstra:1#Wirth:0
Wirth:1#Knuth:1
Dijkstra:2#Knuth:3
Wirth:1#Dijkstra:0
Brasileirinho
8
Flamingo
Sao_Pedro
Bombeiros
Aloha
Coqueiros
Pecadores
Pedro_Cabral
DTB
12
Flamingo:5#Sao_Pedro:0
Bombeiros:3#Aloha:0
Coqueiros:2#Pecadores:1
Pedro_Cabral:1#DTB:1
Flamingo:2#Coqueiros:1
Sao_Pedro:3#Aloha:0
Bombeiros:0#Pedro_Cabral:2
DTB:1#Pecadores:2
Coqueiros:3#Aloha:1
Pecadores:3#DTB:0
Pedro_Cabral:0#Flamingo:2
Bombeiros:1#Sao_Pedro:2
```

Saída

```
Torneio-dos-Programadores
1 - Knuth: 4 pontos, 2 jogos (1-1-0), d.g. 1 (4-3)
2 - Wirth: 4 pontos, 3 jogos (1-1-1), d.g. 0 (2-2)
3 - Dijkstra: 3 pontos, 3 jogos (1-0-2), d.g. -1 (3-4)

Brasileirinho
1 - Flamingo: 9 pontos, 3 jogos (3-0-0), d.g. 8 (9-1)
2 - Pecadores: 6 pontos, 3 jogos (2-0-1), d.g. 3 (6-3)
3 - Coqueiros: 6 pontos, 3 jogos (2-0-1), d.g. 2 (6-4)
4 - Sao_Pedro: 6 pontos, 3 jogos (2-0-1), d.g. -1 (5-6)
5 - Pedro_Cabral: 4 pontos, 3 jogos (1-1-1), d.g. 0 (3-3)
6 - Bombeiros: 3 pontos, 3 jogos (1-0-2), d.g. 0 (4-4)
7 - DTB: 1 pontos, 3 jogos (0-1-2), d.g. -4 (2-6)
8 - Aloha: 0 pontos, 3 jogos (0-0-3), d.g. -8 (1-9)
```

Entrada	1 Desempate_por_nomes 6 Ultimo Posicao6 posicao5 Posicao4 O_vice Campeao 6 Ultimo:1#Posicao6:2 Ultimo:2#Posicao6:1 posicao5:2#Posicao4:1 posicao5:1#Posicao4:2 O_vice:1#Campeao:2 O_vice:2#Campeao:1
Saída	Desempate_por_nomes 1 - Campeao: 3 pontos, 2 jogos (1-0-1), d.g. 0 (3-3) 2 - O_vice: 3 pontos, 2 jogos (1-0-1), d.g. 0 (3-3) 3 - Posicao4: 3 pontos, 2 jogos (1-0-1), d.g. 0 (3-3) 4 - posicao5: 3 pontos, 2 jogos (1-0-1), d.g. 0 (3-3) 5 - Posicao6: 3 pontos, 2 jogos (1-0-1), d.g. 0 (3-3) 6 - Ultimo: 3 pontos, 2 jogos (1-0-1), d.g. 0 (3-3)