De postre, un chuletón

En mi barrio todo el mundo habla de un restaurante que tiene un menú del día muy completo, y a un precio bastante razonable. El menú incluye un primer plato, un segundo plato y un postre. Llevado por la curiosidad, decidí bajar a probarlo ayer. La comida estaba buena, sí, pero lo que más me sorprendió fue la carta con los platos del menú:

MENÚ DEL DÍA

Manzana asada
Espinacas a la crema
Merluza a la plancha
Menestra de verduras
Lasaña
Flan
Chuletón de vaca

¡Vaya lío! Como estaban todos los platos mezclados, me costaba averiguar cuáles servían de primer plato, cuáles de segundo plato, y cuáles de postre. Y no era el único. Por ejemplo, en la mesa de al lado había una familia con un niño que se empeñaba en comerse un chuletón de postre.

Pregunté a la dueña del restaurante el motivo de esta carta tan caótica, y me dijo que preferían ordenar los platos en función de su disponibilidad: al principio aquellos de los que tienen más existencias y al final los de menos existencias. Lo hacen así porque, al parecer, los clientes suelen pedir más los platos que aparecen al principio de la carta, y así no corren el riesgo de quedarse sin aquellos que son más escasos.

Aunque puedo entender los motivos de ordenar los platos de esa forma, sigo pensando que al menos deberían estar agrupados por su categoría. Es decir, todos los primeros juntos, después todos los segundos, y por último los postres.

Supongamos los siguientes tipos de datos:

```
enum class Categoria { Primero, Segundo, Postre };

struct Plato {
  Categoria categoria;
  string nombre;
};
```

Se pide implementar una función con la siguiente cabecera:

```
void ordenar_menu(list<Plato> &platos);
```

La función debe reordenar los elementos de la lista de entrada de modo que todos los primeros estén al principio de la lista, todos los segundos después, y por último los postres. Dentro de la misma categoría, debe preservarse el orden que tenían los platos inicialmente.

Requisitos de implementación:

- No pueden almacenarse los platos en una estructura de datos auxiliar. La función debe realizar los cambios directamente sobre la lista `platos` pasada como parámetro.

- No se puede cambiar el tipo `list` del parámetro por otra implementación del TAD Lista (por ejemplo, `vector`).
- Indica el coste de la función `ordenar_menu` en función del tamaño de la entrada.

Entrada

La entrada consta de una serie de casos de prueba. La primera línea de cada caso indica el número N de platos del menú ($N \leq 20000$). Después vienen N líneas, cada una con el nombre de un plato (máximo 100 caracteres), precedido por su categoría (`1` = Primero, `2` = Segundo, `P` = Postre).

La entrada finaliza con un menú de `0` platos, que no se procesa.

Salida

Para cada caso de prueba debe imprimirse la lista de platos del menú tras la llamada a `ordenar_menu`. El formato de los platos es el mismo que el de la entrada: primero la categoría, y luego el nombre del plato. Tras la lista de platos, debe imprimirse una línea con tres guiones (`---`).

Entrada de ejemplo

```
7
P Manzana asada
1 Espinacas a la crema
2 Merluza a la plancha
1 Menestra de verduras
1 Lasagna
P Flan
2 Chuleton de vaca
3
2 Escalope de pollo
1 Lentejas
2 Sepia a la plancha
0
```

Salida de ejemplo

```
1 Espinacas a la crema
1 Menestra de verduras
1 Lasagna
2 Merluza a la plancha
2 Chuleton de vaca
P Manzana asada
P Flan
---
1 Lentejas
2 Escalope de pollo
2 Sepia a la plancha
---
```

Autor

Manuel Montenegro