El archivo de texto *dicc.txt* que viene a continuación almacena un diccionario representado como una tabla de hashing cerrado.

```
celular:aparato portatil de un sistema de telefonia celular

casa:edificacion construida para ser habitada

posada:establecimiento economico de hospedaje para viajeros
alimento:sustancia ingerida por un ser vivo
```

Todas las líneas son del mismo tamaño y terminan con un fin de línea (caracter '\n'). Se usa el caracter ':' para separar la llave de su valor asociado. Una línea que contiene solo espacios en blanco no contiene ninguna definición.

Escriba en el archivo *consultar.c* un programa que busque una llave en el archivo y entregue su valor asociado. El siguiente es un ejemplo de uso:

```
$ ./consultar.bin dicc.txt casa
edificacion construida para ser habitada
```

En donde el primer parámetro es el nombre del archivo que contiene el diccionario y el segundo parámetro es la llave consultada. En el archivo *consultar.c* Ud. deberá programar la función:

*int main(int argc, char *argv[ ]);*

Requerimientos

- **Debe ser eficiente.** Use fseek para primero buscar la llave en la línea *n* en donde *n* se calcula como *hash_string(llave)* módulo número de líneas del archivo. Si no está en esa línea, búsquela en la siguiente línea y así hasta encontrar la llave o encontrar una línea en blanco, en cuyo caso la llave no está definida en el archivo. El archivo es circular: si llega al final, contínue desde el comienzo.
- Está prohibido leer el archivo completo, salvo si el archivo está lleno.
- Calcule el tamaño de las líneas leyendo solo la primera línea. Se garantiza que una línea no puede tener más de 800 caracteres.
- Calcule el tamaño y el número de líneas en el archivo usando *ftell*.
- Debe entregar el valor asociado a la llave en la salida estándar.

Debe diagnosticar las situaciones de error en la salida estándar de errores. En caso de error termine el programa de inmediato. Estos son los errores que debe diagnosticar:

- No se puede abrir el archivo. Debe diagnosticar este error con *perror*.
- El archivo está vacío.
- El tamaño del archivo o el tamaño de una línea que Ud. tuvo que leer no es consistente con el tamaño de la primera línea. Por ejemplo el tamaño del archivo no es múltiplo del tamaño de la primera línea.
- No se encuentra en el archivo la llave consultada.

El test de prueba incluido verifica que Ud. diagnostique estos errores con exactamente el mismo mensaje que entrega la solución de referencia incluida en formato binario. Para averiguar qué mensajes son, ejecute:

```
$ ARCH=$(arch)
$ bash test-consultar.sh ./prof.ref-$ARCH
```

También puede probar la solución de referencia con un diccionario y llave específica. Por ejemplo:

```
$ ./prof.ref-$ARCH dicc3.txt gnu
```

Instrucciones

Descargue *t4.zip* de U-cursos y descomprímalo. Ejecute el comando *make* sin parámetros en el directorio *T4* para recibir instrucciones acerca del archivo en donde debe programar su solución (*T4/consultar.c*), cómo compilar y probar su solución, los requisitos que debe cumplir para aprobar la tarea (*make run-san*, *make run-g* y *make run*) y cómo entregar su tarea por U-cursos (*make zip*).

Entrega

Ud. solo debe entregar por medio de U-cursos el archivo *consultar.zip* generado por el comando *make zip*. **A continuación es muy importante que descargue de U-cursos el mismo archivo que subió, luego descargue nuevamente los archivos adjuntos y vuelva a probar la tarea tal cual como la entregó**. Esto es para evitar que Ud. reciba un 1.0 en su tarea porque entregó los archivos equivocados. Créame, sucede a menudo por ahorrarse esta verificación. Se descontará medio punto por día de atraso. No se consideran los días de receso, sábados, domingos o festivos.