UNIVERSIDADE DA BEIRA INTERIOR
Departamento de Informática
SISTEMAS OPERATIVOS 2024 Exericico 1

In many constrained environments, for instance a satellite, it is necessary that a product contains a description of itself. Obviously knowing a parts id and manufacturing company (brands) id is not sufficient, one must consult a database to get the actual part and brand details.

The database file in this case is a structured binary file containing an array of records. The database file (example: file.pecas) consists of a header followed by a section detailing the brands (manufacturing companies) then an array of records of type Part (Peca) ordered by id or ordered by name.

```
#define SZ_NAME 16
#define SZ_BRAND 16
typedef struct
{
  int id;
  char name[SZ_NAME];
  int brandId;
  unsigned int weight;
  double price;
} Peca;
#define SZ_PECA (sizeof(Peca))
```

The header consists of 28 Bytes. The first byte 20 bytes are a string, the first character T to indicate the type of ordering of the parts (the character 0,1) to indicate if the parts are ordered by id (T='0') or by name (T='1'). This is followed by 19 bytes for a string description. Next are two binary encoded integers (4 bytes each) the number of parts (NP) and the number of brands (NB).
The next section consists of a database of brands ordered as a set of brands. Each brand is a string of at most 16 characters including the string terminator.

Example (human readable form - is byte 0).

HEADER: 0/1DataBase-PartsV1------6---2
BRANDS: Bosch----------Renault---------

So in this example there are 2 brands (NB=2) and 6 parts (NP=6) where Bosch has brand id 0 and Renault has brand id 1. Below is an example of the "parts" section

Example T=0 (in human readable form)	Example T=1 (in human readable form)
0 px 0 0 0	3 pa 1 3 0.3
1 pd 1 1 1	1 pd 1 1 0.1
2 ph 0 2 2	4 pf 0 4 0.4
3 pa 1 3 3	5 pg 1 5 0.5
4 pf 0 4 4	2 ph 0 2 0.2
5 pg 1 5 4	0 px 0 0 0.0

Your program must be space and time efficient as in this type of environment RAM is at a premium and CPU usage is battery intensive. Therefore, in this exercise you will manipulate the database of parts in a binary file, using open(), close(), read(), write(), and lseek(). You may NOT use the standard library file functions fopen() fread() fwrite() etc... calls. However, you may use the standard library for reading and writing to a terminal (standard input scanf() and output printf()).

The program must accept one command line parameter – the name of the database file.

It is suggested that you create a separate standalone program to create test database files of your choosing.

The program should accept the following options entered via standard input : 1,2,3,4,5 ou 9 with the following semantics.

1 print the header
9 to exit

--When the database file is ordered by ID the options 2,3, and 4 are also valid
2 view a part with id
3 add a new part specifying (in this order) name brand id weight and price – the id can be added automatically.
4 to change a part with id specifying (in this order) a new name brand id weight and price

--When the database file of parts is ordered by part NAME only the further option 5 is valid
5 to view a part with name

The program must be efficient and use a minimal amount of memory. You are advised to not use large static or dynamic arrays in memory or any linear time algorithms. Of course, a separate program for database creation may use dynamic memory and linear loops to create large test files.

<u>Limits</u>

0 < NB <=1000000
0 < NP <= 1000000

<u>Header output</u>

The word "HEADER" followed by the number of parts and number of brands. Tokens separated by a space. Example: HEADER 6 2

<u>Part Output</u>

The word PECA followed by the part id its name the brand id and name then the parts weight and price.

Example. PECA 5 pg 1 Renault 5 5.000000

All values should be wrriten to the output using standard printf format options %lf fpr a double, %d for an integer and %s for a string.

Sample Executions

Sample 1

Example: File db1.pecas with T=0 and the data from the above example.

Input: in.1

```
1
2 0
2 5
2 2
9
```

Execution Example: ./pecas db1.pecas < in.1

Output

```
HEADER 0DataBase-PartsV1 6 2
PECA 0 px 0 Bosch 0 0.000000
PECA 5 pg 1 Renault 5 5.000000
PECA 2 ph 0 Bosch 2 2.000000
```

Sample 2

Example: File db1.pecas with T=0 and the data from the above example.

Input: in.2

```
1
3 newPart 0 10 11.1
4 0 pchange 0 1 1.0
1
2 6
2 0
9
```

Execution Example: ./pecas db1.pecas < in.2

Output

```
HEADER 0DataBase-PartsV1 6 2
6 newPart 0 10 11.100000
HEADER 0DataBase-PartsV1 7 2
PECA 6 newPart 0 Bosch 10 11.100000
PECA 0 pchange 0 Bosch 1 1.000000
```

Notice that in this third example the file db1.pecas will be altered.

Sample 3

Example: File db2.pecas with T=1 and the data from the above example.

Input: in.3

```
1
5 pa
5 px
5 pf
9
```

Execution Example: ./pecas db2.pecas < in.3

Output

```
HEADER 1DataBase-PartsV1 6 2
Enter Part Name: PECA 3 pa 1 Renault 3 3.000000
Enter Part Name: PECA 0 px 0 Bosch 0 0.000000
Enter Part Name: PECA 4 pf 0 Bosch 4 4.000000
```

Notes