Programming assignment 2 - Candy warehouse

A candy netstore Yum'n'Yam has noticed, that its records of the candy stored in its warehouse do not stay up to date. The company needs a small program, which keeps record of selling candy, adding candy to the warehouse, and information about stored candy. Candy is stored in the warehouse in whole boxes. You'll gladly accept this delicious challenge and start designing a program.

The main focus should be the data structure which keeps track of the warehouse contents. Adding and removing boxes of candy from the warehouse records must happen in $O(1)$ time on average. Asking about the amount of specific candy in the warehouse should also be $O(1)$ on average. To make this goal possible, it is allowed that a single on average $O(n)$ operation is allowed occasionally when the warehouse records become "full".

Candy warehouse

Typically the warehouse contains about one thousand (1000) different types of candy. The program has to be able to handle larger (unlimited) amounts, though, it's not allowed to limit the maximum amount of different candies. The candies in the warehouse are divided into candy categories. The categories and their warehouse IDs are in table 1.

Taulukko 1: Candy categories in the warehouse

Category	**ID**
Liquorice	LKS
Chewing gum	GUM
Salmiakki (salty liquorice)	SMK
Miscellaneous	MIX
Chocolate	SUK
Lollipop	TIK
Marshmallow	VHT
Wine gum	WIN

Product code

Each candy has its own unique product code. The code consists of category and storing place, manufacturer ID and product ID. The structure of the product code is following:

$$\underbrace{\text{ABC}}_{\text{category}} : \underbrace{\text{storing place}} : \underbrace{\text{nnnn}}_{\text{manufacturer ID}} : \underbrace{\text{mmm}}_{\text{product ID}}$$

The category is the three letter ID in table 1. Storing place is an arbitrary string containing only numbers and lowercase and uppercase letters between A-Z. Manufacturer ID `nnnn` is a positive integer 1000-9999 and product ID `mmm` is a positive integer 100-999. The parts of the product code are separated by a colon (:).

Warehouse information file

In addition to the product code, the program keeps records of the following things: how many boxes of that product are in the warehouse, shelf code and product name. The number of boxes is a non-negative integer. Shelf code is of form `xx.yy`, where both `xx` and `yy` are integers 1-99. Product name is a string, which can contain numbers, letters A-Z (upper/lowercase), spaces and dashes (-). In the file, each line contains information about one product. So the format of the file is:

```
product code;number of boxes;shelf code;product name
```

i.e.

```
abc:abcdefg:nnnn:mmm;lkm;xx.yy;abc def
```

For example:

```
LKS:34X287a9:1234:123;152;42.2;Panda Lakupala
GUM:137856XQJ:2345:222;500;58.92;Jenkki Pro Freshmint
VHT:FtG56Rt:9876:321;5;12.1;Jet-Puffed Marsmallows
SMK:098765retfg:1212:965;1234;8.97;Tyrkisk Peber
...
```

How the program works

The commands recognized by the program are listed in table 2, with command parameters and descriptions. In the parameters, <code> is a product code described earlier, <n> is an integer, <p> is a shelf code, and <k> is a string (A-Z, a-z, 0-9, space, dash).

Parts provided by the course

The main focus on this assignment is the implementation of the data structure. Implement your data structure to file datastructure.cc and datastructure.hh. Template for these files are given in the version control, including the public interface for the class (which must not be modified).

The main program is again given by the course and MUST NOT be modified. It read the input and the record file, and calls member functions of your datastructure class. The main program is used to test the data structure.

As before, several tests (testing both functionality and performance) are also given.

Error situations

The main program deals with error in input data. Some error messages come from the data structure, and those error codes are described below. The definitions of the error codes are found in datastructure.hh.

- If there is not enough specified candy in the warehouse, prints out error code NOT_ENOUGH_CANDY, how many boxes are available, and the shelf code.
- If specified candy is not available at all in the warehouse, prints out error code NOT_AVAILABLE.

Example run

```
> R varasto.txt
> A LKS:34X287a9:1234:123 42.2 50 Panda Lakupala
> A MIX:9ikju76t:3679:999 1.11 25 Re-mix
> A SUK:4567uij98:8787:324 93.93 124 Suffeli
> F LKS:34X287a9:1234:123
Panda Lakupala 202 42.2
> F WIN:9o8iu7y6t5r:5654:236
Product not in warehouse.
> D SUK:4567uij98:8787:324 22
Amount: 102 Shelf: 93.93
> D MIX:9ikju76t:3679:999 58
Not enough candy available.
Amount: 25 Shelf: 1.11
> C
6 different kinds of candy.
> Q
```

Limitations

The standard library of C++11/14 can mostly be used. Use of associative containers `unordered_map`, `unordered_set`, `map` ja `set` and their multi-

map/set variations is not allowed. For sequence containers, `list` is not allowed. Other sequences (vector, deque, array, C array) are allowed, as long as their size remains constant (i.e., they are created to a certain size, and the size is not changed afterwards).

Taulukko 2: Commands understood by the program

Command	**Description**
R <F>	Reads warehouse records from file <F> and adds them to the data structure. If the filename is not given, reads records from file varasto.txt.
A <code> <p> <n> <k>	Adds <n> boxes of candy with product code <code> and name <k> to shelf with shelf code <p>. If candy <code> is already in the warehouse, new boxes are added to the already existing information. You can assume that the shelf code and the name stay the same for the same product code, if it is added several times.
D <code> <n>	Decreases the amount of candy <code> byt <n> boxes. Prints out the remaining number of boxes and the shelf code. If there is not enough candy <code> in the warehouse, prints out error code NOT_ENOUGH_CANDY, how many boxes are available, and the shelf code. If candy <code> is not available at all in the warehouse, prints out error code NOT_AVAILABLE. If the number of boxes for a candy become zero, removes the record for that candy from the data structure.
F <code>	Finds out if product <code> is found in the warehouse. Prints out product name, the number of boxes available, and shelf code. If candy <code> is not available at all in the warehouse, prints out error code NOT_AVAILABLE.
C	Prints out how many different candies there are in the warehouse.
E	Empties out the data structure.
Q	Quits the program.