Programming Assignment 3
Inventory Management

CPTS 223 Advanced Data Structures

Accept this assignment by clicking on the following GitHub Classroom link:
https://classroom.github.com/a/pAwGQi_N
I strongly recommend accepting the assignment **immediately** and doing at least one test push! Work with the TAs if you face any issues. We will not entertain last-minute tech support requests for GitHub Classroom. Accepting the link will create a repository for you and the repository contains skeleton code for the assignment.

For this assignment, please build a command line REPL system that allows the user to query the Amazon inventory. Please see the skeleton template for the REPL. The REPL should support the following commands.

1. find <inventoryid> - Finds if a product exists whose 'Uniq id' matches the 'inventoryid'. If it exists, print the details of the product. If not, print 'Inventory/Product not found'.
2. listInventory <category_string> - Lists the 'Uniq Id' and 'Product Name' of all products whose 'Category' matches 'category_string'. If the category_string does not match any 'Category', then print 'Invalid Category'.

If interested, you can implement more commands to improve your application, though there are no bonus points. At a minimum, the application should support the two commands above. The repository contains the skeleton code for the assignment.

The Dataset

We have downloaded the Amazon product dataset CSV and will distribute the file via Canvas to make it convenient for you. If needed, you can download the latest copy directly from Kaggle via the link below:
https://www.kaggle.com/datasets/promptcloud/amazon-product-dataset-2020

Notes on the dataset:

1. Please note that a product can belong to multiple categories. For example, the product "VTech Twist and Hug Koala Rattle" belongs to "Toys & Games", "Kids' Electronics", and "Electronic Learning Toys". In the CSV file, the '|' symbol is used as the delimiter for categories.
2. Sometimes, data might be missing. For example, the product "Woodstock—Collage 500 pc Puzzle" is missing category data (i.e., the category column is empty). In such cases, you can use "NA" as the category to indicate that the category information is not available.
3. You may need to do additional cleanup. Please make your own decisions and document them in the readme file (readme.md).

Instructions on the Implementation:

1. We are going to leave most of the design decisions to you.
2. We need a functioning application that highlights the following.
    a. Your ability to identify the correct data structure(s) for the problem.
    b. Your ability to implement the data structure container(s) using templates.
    c. Your ability to utilize the container to implement the application.
    d. We would also expect you to follow good/professional programming practices.
3. First, you must decide on the data structure(s) best suited for solving the problem. But you can't use out-of-the-box STL containers. Instead, you need to **implement the container from scratch**.
    a. First, write the container from scratch. For this, you should write them as class templates.
    b. Once you test your container on simple problems to verify that it is working as expected, you can use it for the Amazon Inventory System.
4. If you decide to use a hashtable as one of your data structures, you can use **std::hash** or other openly available hash functions. In other words, you don't have to implement any complicated hashing like murmurhash2 by yourself. Just use what is already available.

Software Testing

- Testing catches bugs early and validates that your data structure works correctly before building the application on top of it, saving significant debugging time and ensuring correctness.
- Testing is fundamentally about creating a reasonable subset of **(input, expected output)** pairs. You cannot test every possible input, so you select representative cases covering normal usage, edge cases, and error conditions.
- You **must implement test functions** using **cassert** to test all **operations** of your container(s) with simple data before integrating them into the inventory application. Each test should use at least **two test cases**, one for the normal case and one for the edge case.
- Document your testing approach in your README.md file, including what test cases you created and why you chose them.

Note: We will cover cassert and testing techniques in class early next week.

Walkthrough Video (10 pts)

Record and submit a short code walkthrough video (walkthrough.mp4). In this video, you must demonstrate using the **debugger**:

1. How does inventory *find* work in your implementation?
2. How does the *list_inventory* work in your implementation?

Points Breakdown

Component	Points
Correct selection of data structure(s)	10
Correct implementation of container(s) using templates (includes support for insert, find, etc.)	20
Successful `find <inventoryid>` command - must utilize your container	15
Successful `listInventory <category>` command - must utilize your container	15
CSV data processing and loading	10
Product/Inventory class design	10
Testing functions – along with clear explanation of test cases in README	5
Use of Makefile, proper modularization and code cleanliness	5
Video code walkthrough - using debugger	10
Total	**100**

Submission Guidelines

- Follow proper programming practices: split your solution meaningfully into multiple .cpp and .h files, rather than putting everything in main.cpp.
- Use C++11 as the standard.
- Include a Makefile. Ensure your Makefile can build the project cleanly.
- Push your code to your GitHub Classroom repository created by accepting the link in this document.
- Upload the following to Canvas as your official submission

1. repo.txt containing only the HTTPS URL of your GitHub Classroom repository on a single line.
2. walkthrough.mp4 screen recording using a debugger.