CS 2336 – PROJECT 3 – Redbox Inventory System

Pseudocode Due: 10/20 by 10:00 PM

Project Due: 11/5 by 10:00 PM

KEY ITEMS: Key items are marked in red. Failure to include or complete key items will incur additional deductions as noted beside the item.

Submission and Grading:

- All project deliverables are to be submitted in eLearning.
- Zip all of the source files into a single zipped file
    - Make sure the zipped file has a .zip extension (not .tar, .rar, .7z, etc.) (-5 points)
    - Please review the submission testing information in eLearning on the Course Homepage
- Projects submitted after the due date are subject to the late penalties described in the syllabus.
- Programs must compile using gcc 7.1.0 or higher with the following flags enabled
    - -Wall
    - -Wextra
    - -Wuninitialized
    - -pedantic-errors
    - -Wconversion
- Each submitted program will be graded with the rubric provided in eLearning as well as a set of test cases. These test cases will be posted in eLearning after the due date. Each student is responsible for developing sample test cases to ensure the program works as expected.
- **Type your name and netID in the comments at the top of all files submitted.** (-5 points)

Objectives:

- Implement a binary search tree class in C++.
- Utilize a binary search tree
- Perform simple input validation with string functions

Problem: Redbox needs a program to track inventory and to generate a report for their DVD rental kiosks. Given a log of transactions including renting and returning DVDs as well as adding and removing DVD titles, the program will need to process each transaction and create a report after all transactions have been processed. The generated report will list all DVD titles stored in the kiosk as well as how many of each disc are in the kiosk.

Pseudocode: Your pseudocode should describe the following items

- Main.cpp
    - List functions you plan to create
        - Determine the parameters
        - Determine the return type
        - Detail the step-by-step logic that the function will perform
    - Detail the step-by-step logic of the main function
- Binary search tree functions

    - Insert
    - Search
    - Delete
    - Any other functions you plan to add to the class

Details

- The inventory will be held in a binary search tree that you will create
- Use the DVD title to determine node placement in the tree
- The binary tree will be seeded with an inventory file
- Once seeded, the program will parse a transaction log to update the inventory
- There are five possible transactions
    - Add new title
    - Add copies for an existing title
    - Remove copies of an existing title
    - Rent a DVD
    - Return a DVD
- Add new title
    - Create a new node and insert it into the tree
- Add copies of an existing title
    - Find the title in the tree and increase the number of available copies by the amount listed
- Remove copies of an existing title
    - Find the title in the tree and reduce the number of available copies by the amount listed
    - If number available is zero and no copies are rented out, delete the node from the tree
    - There will not be more copies removed than available.
- Rent a DVD
    - Reduce available amount by one and increase rented amount by one
- Return a DVD
    - Increase available amount by one and reduce rented amount by one

Classes

- Node
    - Members
        - Title (string)
        - Available (integer)
        - Rented (integer)
        - Left (node pointer)
        - Right (node pointer)
    - Methods
        - Overloaded constructor
        - Mutators
        - Accessors
- Binary Search Tree
    - Binary Search Tree class must be templated (-5 points if not)

- Members
  - Root (node pointer)
- Methods
  - Mutator
  - Accessor
  - Insert (recursive) (-5 points if not)
  - Search (recursive) (-5 points if not)
  - Delete
  - Other methods that are necessary to interact with a binary search tree
    - Remember methods should be generic enough to be used on a binary tree regardless of the problem.

User Interface and Input: There is no user interface for this program.

Input will be given in two files, `inventory.dat` and `transaction.log`. The program will read `inventory.dat` first. This will create the binary tree. Each line (except the last line which may not have a newline character) in `inventory.dat` will be formatted as follows (NOTE: <text> represents a variable value):

`"<title>",<quantity available>,<quantity rented>`

After processing the inventory file, begin processing `transaction.log`. Each line of the file should follow one of the following formats (Note: the last line may not end with a newline character):

- `add "<title>",<number to add>`
- `remove "<title>",<number to remove>`
- `rent "<title>"`
- `return "<title>"`

The transaction file may contain errors due to network disruptions from the main server. For each line in the transaction log, validate that it follows one of the formats listed above. If it is the correct format, process the transaction. If the line is invalid, write the line to an error file (as described below). All numbers are expected to be integers. To be valid, the line must follow the format exactly. Also, do not assume that a title in the transaction log will be in the tree.

Output: A file named `error.log` will be created if any lines of `transaction.log` are invalid. `Error.log` will contain all invalid entries of the transaction file.

At the end of the program, create a formatted report to display each title, the number of copies available to rent for that title as well as the number of copies that are currently rented. The titles should be listed in alphabetical order (without the double quotes). The report should be arranged in three formatted columns that line up the data nicely:

- Title
- Copies available
- Copies rented

Write the report to a file named `redbox_kiosk.txt`.