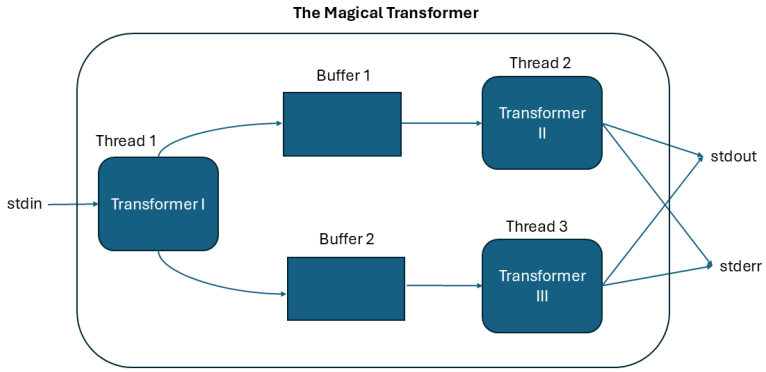
**EEL 4732/5733 Advanced Systems Programming**

**Assignment 2**

In this assignment, you are going to implement a magical transformer program that generates some reports about real-estate transactions. It uses three threads, Transformer I, Transformer II, and Transformer III, described below to process data and to provide a flexible way for users to choose which report appears on which output stream.

You can reuse your implementation of the transformers from your solution for Assignment 1. However, you need to realize that you will need to include the implementations of all the transformers in one single program as in this assignment the magical transformer is a single process application with multiple threads.

The software architecture of the magical transformer will be as shown in the figure below:




The magical transformer will create three **threads** to perform the functionality of Transformer I, Transformer II, and Transformer III and will create one buffer to enable inter-thread communication between Transformer I and Transformer II and another buffer for communication between Transformer I and Transformer III. You will be implementing the producer consumer problem for each of these inter-thread communications. Your solution must implement proper synchronization among the threads to avoid data races and to optimize CPU usage by putting the threads into a waiting state when it is needed. Also, your solution to the producer-consumer problem will handle finite data so that threads can determine when they should terminate. You are expected to implement your multithreading solution using the Pthreads API we discussed in class.

The magical transformer will receive the input from the standard input. The grammar for the command line options will be

[functionality:stream_name]*

where stream_name can be stdout or stderr and functionality can be any of {agent_rating, agent_performance, state_rating, state_performance}.

Absence of any command line arguments means that NO output will be generated and only a message will be displayed:

$ ./magic_transformer
No output requested, have a nice day!

When there are command line arguments then some output may be produced as long as there are no inconsistencies among the command line arguments as explained below.

The following command

$ ./magic_transformer agent_rating:stdout state_performance:stderr

will output the agent rating information on the standard output and the state performance information on the standard error for the magic transformer process.

It is acceptable to bind the same stream to multiple types of output, but it is not acceptable for the same type of output to appear on multiple streams. So, the following one is acceptable:

$ ./magic_transformer agent_rating:stdout agent_performance:stdout state_performance:stderr

in which case it is normal to see data from different type of functionality to get mixed on the standard output stream.

The following should not generate any output and generate an error message:

$ ./magic_transformer agent_rating:stdout agent_performance:stdout agent_rating:stderr
Error: agent_rating is directed to more than one output stream!

Please test your code on a Linux system (Ubuntu 20 and newer versions are recommended).

**Important Notes:**

1) The magical transformer program must be implemented as a single-process and a multi-threaded application.
2) Implementations of the magical transformer program that do not make effective use of the Pthreads API will not get any credit.
3) Make sure that all the programs detect the end of file character while reading from the standard input.

**Description of Transformer I, Transformer II, and Transformer III**

**Transformer I**: Reads the real-estate agent data from the standard input. The data is in the following format (on each line):

*Agent Name, Agent Id, Transaction Id, Real Estate Location, Original Price, Sale Price, Customer Rating*

where,

Agent Name: 10 characters max
Agent Id: Exactly 5 characters, in the range of [00000-99999]
Transaction Id: 10 characters, in the range of [0000000000-9999999999]

Real Estate Location: Two letter state code
Sale Price/Original Price: Max 14 characters, in the range of [0-999,999,999.00]
Customer Rating: Max 3 characters, in the range of [0-5.0]

It transforms the data into two streams; the price/performance information is output on the standard output and the rating information is output on the standard error.

The price information will be in the following format:

*Agent Name, Agent Id, Transaction Id, Real Estate Location, Sale Price, Loss/Gain*

where the existing fields will be formatted as described above and Loss/Gain represents Sale Price – Original Price and is formatted the same way as the Sale Price except it may have the sign symbol (– /+) (so 15 characters max).

The rating information will be output in the following format:

*Agent Id, Real Estate Location, Customer Rating*

where all the fields will be formatted as described above.

You need to ensure that both outputs include all the relevant data in the input.

**Transformer II:** Reads the price information (as described above for Transformer I) from the standard input and transforms it into two types of outputs: 1) the agent's maximum performance data to appear on the standard output and 2) the state market data to appear on the standard error.

The agent's maximum performance data will be in the following format:

*Agent Name, Agent Id, Sign Adjusted Maximum Loss/Gain*

where all the fields will be formatted as described above and the Sign Adjusted Maximum Loss/Gain represents the sign adjusted maximum value among the absolute values of loss and gain values for all the real estate sold by the agent across all states. For example, the sign adjusted maximum value for (10,100,-10,-5000) will be -5000 and for (10,100,-10,-50) it will be 100.

The state market data will be in the following format:

*Real Estate Location, Sign Adjusted Maximum Loss/Gain*

where all the fields will be formatted as described above and the Sign Adjusted Maximum Loss/Gain field will represent the maximum loss and gain values of all real estate transactions performed within the state by all the agents.

**Transformer III:** Reads the real estate agent rating information (as described above for Transformer I) from the standard input and transforms it into two types of outputs: 1) the agent's maximum rating data to appear on the standard output and 2) the state's maximum rating data to appear on the standard error.

The agent's maximum rating data will be in the following format:

*Agent Id, Maximum Customer Rating*

where Maximum Customer Rating will represent the maximum of all the ratings received by the corresponding agent as denoted by the Agent Id across all the states in which the agent sold real estate.

The state's maximum rating data will be in the following format:

*Real Estate Location, Maximum Customer Rating*

where Maximum Customer Rating will represent the maximum of all the customer ratings within the state denoted by the *Real Estate Location* field for all the agents that sold real estate within that state.

**Note:** Even though you will implement each program by receiving their input from the standard input, you can still provide the input in a file using the input redirection operator <. Similarly, you can use the output redirection operator > to write the output to a file. So, you can test each of the programs you implement as follows (1 denotes standard output and 2 denotes standard error):

$ ./transformer1 < input.txt 1 > performance_data.txt 2 > rating_data.txt

$ ./transformer2 < performance_data.txt 1 > agent_performance_data.txt 2 >state_performance_data.txt

$ ./transformer3 < rating_data.txt 1 > agent_rating_data.txt 2 > state_ rating_data.txt

Here are some test cases for the magical transformer:

$ ./magical_transformer stdout:agent_rating < input.txt 1 > m_ agent_rating_data.txt

Check if m_ agent_rating_data.txt and agent_rating_data.txt are the same using the diff command:

$ diff m_ agent_rating_data.txt  agent_rating_data.txt
(No differences should be reported!)

$ ./magical_transformer stderr:agent_rating  < input.txt 2 > m_ agent_rating_data.txt

Check if m_ agent_rating_data.txt and agent_rating_data.txt are the same using the diff command:

$ diff m_ agent_rating_data.txt  agent_rating_data.txt
(No differences should be reported!)

The TA will test your code assuming that the magical transformer accepts its input from the standard input and writes to the standard output. Your implementation should not rely on the existence of specific input files. So, please make sure to test your code as suggested above. Otherwise, grading of your submission will be delayed.

**Submission:** Submissions will be through the ELearning portal by the due date. You should submit your source code (in C/C++) for the three programs along with a Readme file and preferably a Makefile to show the details of compiling your code.

**Late submission policy:** For every 6 hours late, 10% of the total score will be deducted.