Assignment W13 Brandon Gopar

Study Plan

- Topic Chosen: Linked List
- Reason: I chose linked lists because they are foundational in computer science and are used in many real-world applications. I wanted to practice manipulating dynamic structures and pointers in C++ through a project with practical relevance.
- Learning Goals:
    - To figure out a real-world application where it can be implemented
    - Understand how to implement a singly linked list in C++
- Resources
    - https://www.geeksforgeeks.org/cpp-linked-list/
    - https://opendsa-server.cs.vt.edu/OpenDSA/Books/CS3/html/ListLinked.html

What was learned

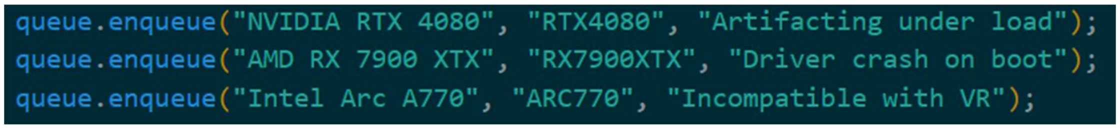
- Concept Summary:
    - The linked list made stores information from a customer regarding issues with their GPU. It gets the name, model, and issue then stores these into a ticket (node). These nodes reference the next node in a list and are connected via pointers. Using the linked list I was able to use dynamic memory allocation, given that it is unknown the number of elements needed from the beginning. Using this in a business would help streamline customer support issues.
- Insights:
    - Pointer management is very important
    - Challenging part was handling an empty list and updating the list
    - Putting the information into a linked list was the easier part
- Time Complexity
    - Enqueue – 0(1)
    - Dequeue – 0(1)
    - Display – 0(n)

Implementation Summary

- Approach:
    - I created a Ticket Node struct to hold ticket information (GPU name, model ID, issue, and timestamp). A Ticket Queue class manages the

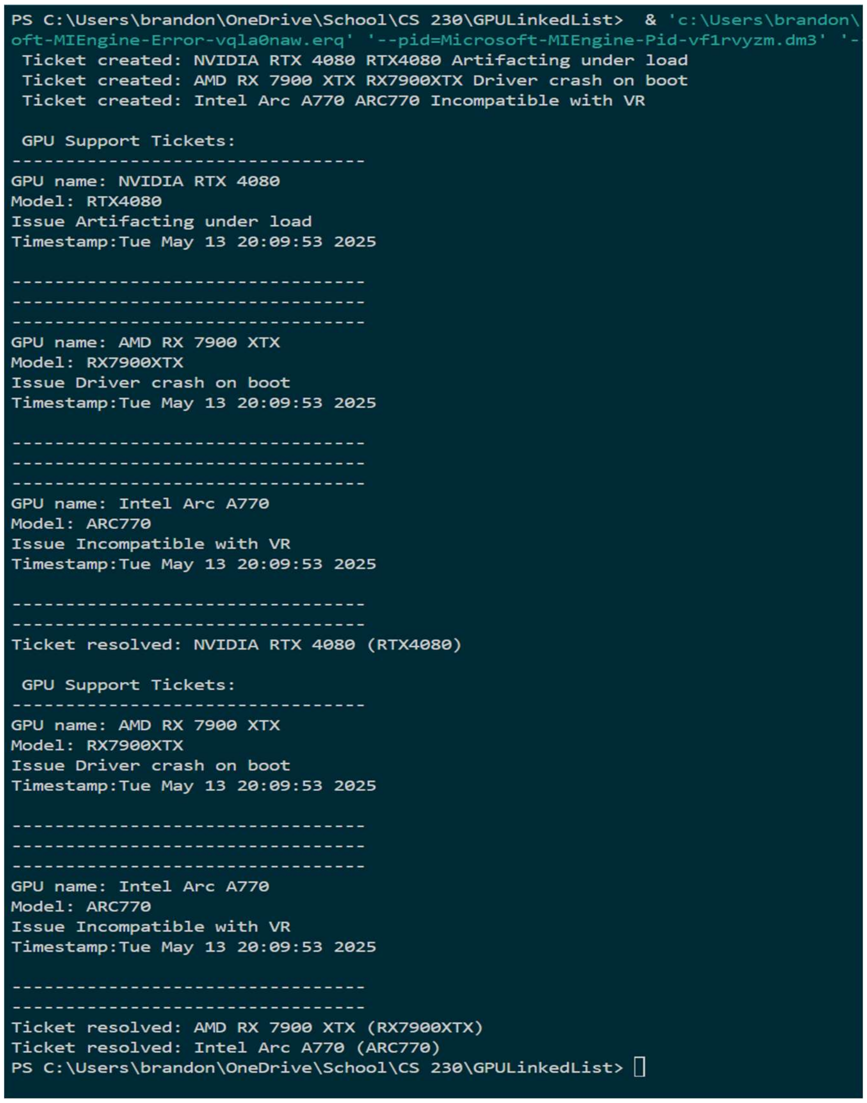
list with enqueue(), dequeue(), and display() methods using pointers to track the head and tail.

- Key Files/ Functions:
    - Main.cpp – Contains the program
    - TicketNode – Struct that holds data and a pointer to the next
    - Enqueue – Adds a ticket to the end of the queue
    - Dequeue – Removes a ticket from the front
    - Display – Prints all active tickets
- Sample Input & Output
    - Input



```
queue.enqueue("NVIDIA RTX 4080", "RTX4080", "Artifacting under load");
queue.enqueue("AMD RX 7900 XTX", "RX7900XTX", "Driver crash on boot");
queue.enqueue("Intel Arc A770", "ARC770", "Incompatible with VR");
```

- Output



```
PS C:\Users\brandon\OneDrive\School\CS 230\GPULinkedList>  & 'c:\Users\brandon\
oft-MIEngine-Error-vqla0naw.erq' '--pid=Microsoft-MIEngine-Pid-vf1rvyzm.dm3' '-
 Ticket created: NVIDIA RTX 4080 RTX4080 Artifacting under load
 Ticket created: AMD RX 7900 XTX RX7900XTX Driver crash on boot
 Ticket created: Intel Arc A770 ARC770 Incompatible with VR

 GPU Support Tickets:
---------------------------------
GPU name: NVIDIA RTX 4080
Model: RTX4080
Issue Artifacting under load
Timestamp:Tue May 13 20:09:53 2025

---------------------------------
---------------------------------
---------------------------------
GPU name: AMD RX 7900 XTX
Model: RX7900XTX
Issue Driver crash on boot
Timestamp:Tue May 13 20:09:53 2025

---------------------------------
---------------------------------
---------------------------------
GPU name: Intel Arc A770
Model: ARC770
Issue Incompatible with VR
Timestamp:Tue May 13 20:09:53 2025

---------------------------------
---------------------------------
Ticket resolved: NVIDIA RTX 4080 (RTX4080)

 GPU Support Tickets:
---------------------------------
GPU name: AMD RX 7900 XTX
Model: RX7900XTX
Issue Driver crash on boot
Timestamp:Tue May 13 20:09:53 2025

---------------------------------
---------------------------------
---------------------------------
GPU name: Intel Arc A770
Model: ARC770
Issue Incompatible with VR
Timestamp:Tue May 13 20:09:53 2025

---------------------------------
---------------------------------
Ticket resolved: AMD RX 7900 XTX (RX7900XTX)
Ticket resolved: Intel Arc A770 (ARC770)
PS C:\Users\brandon\OneDrive\School\CS 230\GPULinkedList>
```

Reflection

This topic and its subsequent subtopics like doubly and circular linked lists are very useful in programming. Their implementation in a business as shown in this example can reduce waiting times and streamline technicians to rapidly understand the problem. I can see myself using this concept for real-time task scheduling, undo/redo features, or memory-efficient data structures in embedded systems. Implementing it in a context that mimics real-world support systems made it even more engaging.