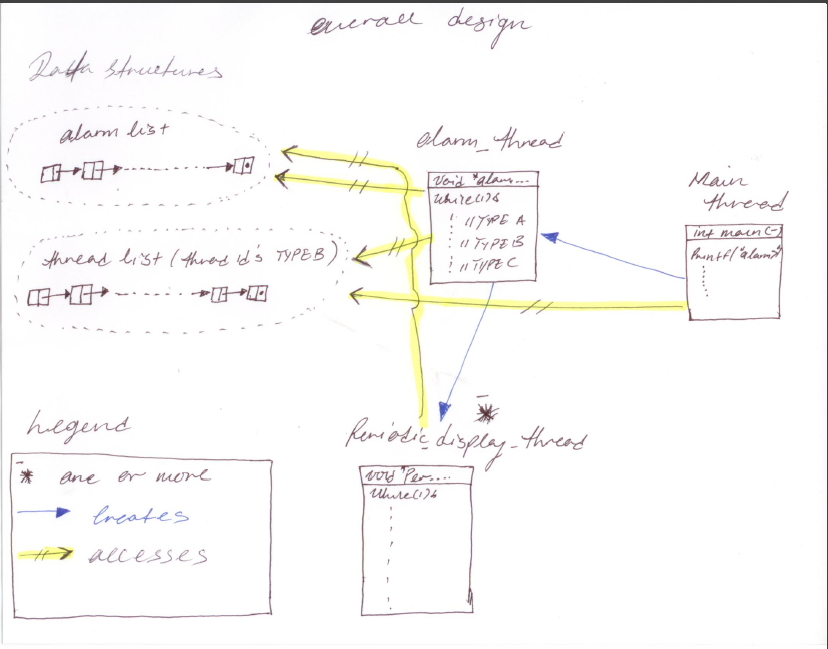
## Design challenges

* Giving prior experience with Assignment 2, the programing aspect of Assignment 3 was not overly complex. There were a lot of bugs during the development process mostly due to the sheer size of the program.
* The most challenging part of this program’s development was strategically placing mutexes and semaphores to allow for fluid data access by the threads.
* Another challenge was figuring out how to make a Type B alarm of say message type 4 print messages of said type without adding the time of the first message to the next one. Example: let’s assume we have 3 type A alarm requests of message type 4 (with times 3, 6 and 1 second respectively) in the alarm list. When the first message prints after the 3rd second, the second message would print 6 seconds after the 1st message was printed and the 3rd message 1 second after the 2nd message was printed. Which means that the second message prints every 9 seconds and the 3rd one every 10 seconds. Then the first type 4 message would printing again 13 seconds after it was initially printed. Adding more alarms with different types would make this problem even more prominent.
* The method for “waiting” was adapted from assignment 2 but it didn’t work properly so we modified it.

## Overall Diagram of System

There should be a scanned copy of the hand drawn diagram in the repository. Please make this in Visio of something that can make good diagrams.



## Implementation

So this program is similar to assignment 2 in that it creates threads to display different messages if different types at specified time intervals. The major difference is that this one puts all the alarm request in a singular list and allows for threads to be created and terminated on the fly. Because threads are being created, in order to terminate them given the conditions stated in the requirements, we had to implement a data structure to store the thread ids. We did this in assignment 2 but not as efficiently as we have in this current assignment.

A considerable amount of the code is helper code. This made the implementation if the threads a lot easier and les crowded.

You will find that the comments / documentation is done in such a way that it sections the code making it easier to understand what does what. The section numbers from the requirements are also present in the code.

You guys can include the method headers and descriptions for the helper code only.