**Effectiveness & Approach**

The design and implementation of Program 2 for me was largely successful. I believe that a majority of my intentions with the program were executed well and preformed as intended. In regard to effectiveness, I believe that some portions were effective and productive in their execution and others less so. In general, my main goal was to create the program to utilize three different communication applications and manage them with an Abstract Base Class. This goal was accomplished, and I believe that the program is effective in this area.

My design was to have one Abstract Base Class, three derived classes that represented the different applications, a user class to track account information as well as 4 different node classes and a list class to manage them. In terms of cohesion, this design made implementing these classes to work well with my data structure was one of the smoothest aspects of my program. However, I would say that this design required a fair amount of time to be spent on data structure creation and management due to the 4 separate node classes. This is most likely my least effective aspect of my design. It preforms its job well, but I believe approaching the inheritance and structure of node classes could have reduced to number of classes and shortened the amount of code needed.

My approach was to utilize the pure virtual functions in the Abstract Base Class to send and receive messages for the program. While this proved to be challenging, I believe the ability to house objects from different derived classes in a single data structure lent itself well to this program and enabled to the program to be more versatile. I chose to house the messages of each of the application classes in their own Linear Linked Lists. The data structures are perfect for a message based program as adding to the front of the list is a great visual representation of how messaging works. To create a variety among the messaging applications I chose to alter the contents of each classes messages, this gave each of the applications a more unique design.

Finally, I wanted to implement the use RTTI as well as a threaded discussion into the same segment of my program. I chose to do this by using dynamic casting to create a threaded discussion in one my discord Linear Linked List node. I accomplished this with the use of another Linear Linked List inside of the discord list. The user could respond to a singular message and continue the thread as they saw fit. I believe a different design of this feature would have enabled me to expand on this feature and use it more within my discord class, but for the sake of time management, I implemented a more simplistic design for this aspect of the program.

**Object Oriented Programming**

In terms of object oriented programming, I believe that I more successful with this program than those prior. I felt that I was able to create a strong hierarchy throughout my classes and paired the classes that complemented each other well. Initially I was going to simply hold a character array in the Linear Linked List of communication pointers to represent the account name. I was able to identify this information as a similarity amongst all of the different accounts and decided to transform this into it’s own user class. This standardized the process of creating an account, as well as provided the rest of the program the ability to access the account information and greatly improved this user’s experience.

If possible, I would have altered my design with my node classes to enable a stronger relationship with my list and application classes. While they worked as intended, I believe a different implementation of their derivation would have allowed me to reduce the work of my application classes and allowed them to more directly work with the data structure, instead of needing to received and send data.

**Changes**

The largest change I did was to change where my list objects existed within in my data structure. Initially I struggled with where to keep this object so my list class could manipulate my Linear Linked Lists of messages. I experimented with passing the object through arguments to the virtual functions. This was largely successful and enabled an easy implementation, but I believed was not inline with object oriented programming. I decided to alter the design and utilize a pointer in my communication pointer list. This “has a” relationship paid dividends and I was able to utilize my previous code with only some slight alterations.

**Efficiency**

The efficiency of my code was mostly positive. I believe my data structures and messaging applications work fluidly. They create, manage, and destroy the content of the message efficiently using recursion and are simplistic in their design. I was able to complete the project with no memory leaks and without cutting any corners for the programs design. I do wish that I could have implemented the code with a different design than having four node classes, this would have reduced the overall amount of code as well as reduce the amount of hours needed to complete it.

**Conclusion**

I believe that I could have approached the programming assignment largely the same. If the intention of the program was to experience using an Abstract Base Class as well as a dynamic binding, I believe I have gained a much larger understanding of these concepts and improved my ability to use object oriented programming.