Thomas McIntosh

Group 5

Assignment 7 Comparisons

Thomas McIntosh compared to Joel Adriance

Neither mine nor Adriance's program worked properly in all situations. My code returned to number of modes instead of returning the value of the modes. His program would properly work most of the time but it did have an error were it would return the incorrect mode if the sorted array contains elements that occur only once appeared sequentially. For testing he corrected his error because it was a simple single line error where he had left off an else statement. With his corrected code it was exponential faster than any of the programs in the group. The reason his program was so much faster was because his code sorted the elements of the array and then compared the values as they appeared in the array to find the frequency count. My code found the matching elements by comparing every element in the array to ever other element in the array. Adriance's program found the modes by counting each group of matching elements, then based on their value would do various actions. If the value pushed into the array was as common as the highest frequency found at the time, or if it was more common it would restart the result array and push the value in it. Finding the elements of the highest frequency is where my code was incorrect my program counted and found the elements of highest frequency, but failed to identify and return them. As far as code readability goes I believe after reviewing the group's codes that while mine may could have used some more comments Adriance's was over commented to the point where the code became hard to read. It is hard to claim that my code did anything better than his since my code would never return the right values, and there are certainly things to be learned from his code even though his also contained an error affecting the results. The fact that he sorts the array first to group the elements that are equal allows his program to run more efficiently. This also allowed his vector to be sorted upon creation allowing his program to not have to sort the vector.

Thomas McIntosh compared to Krunoslav Peric

Peric's program worked properly in all situation and always returned the right result. Things that Peric's code did that stood out was that we declared the array as a constant. Because his array was constant it did not create a local array to store the temporary calculations. Peric found matching elements to count frequency the same as my code did by comparing every element in the array to ever other element in the array. His code only tracks a single value representing the currently highest frequency found while iterating and counting matches. This allows his code to avoid storing an extra array to save on RAM. Once his code has found the highest frequency he then runs through the array again and pushing back any values that have a frequency equal to the highest frequency. The values are then stored in a vector and then sorted and returned to main. The fact that he had to run through the array twice instead of storing them in a vector as they were found slowed down his code making his the slowest out of the groups for large arrays. I do feel that his code was commented very well making his code easy to read and follow. Obviously since his code worked and mine did not his is superior in execution. Specifically the fact that he made the array a constant is a good idea to save memory. I still believe that some of my methods would have been better had my code worked. For example, his method of running through the array as he did was clearly slower than other methods in testing. I think that including a statement to specifically handle if the array only contained on element would have also been an improvement over his code.

Thomas McIntosh compared to Michael Twomey

Twomey's code also worked in all situations. Peric had the overall best code in group. His code was the fastest of the working codes. He was the only code in the group to utilize parallel arrays. He stored the number of occurrences for each value in a parallel array of equal size to the array argument, then he would iterate through the array to find the highest frequency. Because of his use of parallel arrays he was able to have his code shorter than that of the rest of the group, this in tandem with having concise comments made his code very readable. To find elements that have the highest frequency his code would check each element in his parallel array to see if count matches the highest frequency. The values would then be retrieved using the identified indices with the values of the array and then pushed into the results vector. Once the vector was formed it was then sorted. While the use of parallel arrays worked in this situation they can be harder to deal with since whenever you make a change you have to remember to make that change in multiple places. Because of this I not have used this method.

Learning and Improvements:

Since my code did not return the correct results it has been very hard to compare and contrast, because my code is clearly inferior in its original condition. I did rewrite my code once I found out my error, so I now know what I did wrong and will use that knowledge while discussing the other codes and all future codes. My original code replaced the values in the array with their frequency so it would print the frequency of the modes instead of their values, so for example if the proper result was 3, 5, 8 where each one appeared 2 times in the array my code would return, 2, 2, 2. This was clearly a major error in my coding, but it did show me how the use of const, like Peric used, on an array where we do not want to change the values would have been very beneficial. I believe that sorting the array at the beginning, like Adriance did. allowed for the fastest run time because the code would not have to go through the entire array each time it went to another element to find the frequency. This was not apparent in the testing before turning in the project, but when dealing with large arrays it becomes very evident how much time it saves. While Twomey had the best overall code in the group I still would still not use parallel arrays even though it worked for him. My three group mates had very different methods to find the exact same result but their methods were not equal. This showed me that not only is it important to get the proper result it also important to find the most efficient way to get to the result. Peric's being so much faster than the other two showed how a small change in code can have a major change in the overall result in the efficiency of the program. Having these group projects being the first time I have had to read others code that I have not seen before has shown me how important it is to have clearly named variables and clearly written comments. Also it is important to also pay attention to what is required, my original code that I wrote converted the array into a vector in the main and then used the vector as a parameter into my class. This clearly would not work since the main was provided and the class had to accept an array as a parameter, because of this I scrapped that code. By the time I realized my error I was not able to get the new code work properly in time. I also learned that it is important to check your code and requirements repeatedly and to always give ample time for testing and writing. Overall this group project has helped me in my code writing.