Ethan Sutton

Anthony Rohloff

Mini Project 1

Task 1:

Task 1 is just implementing a simple walk forward insertion scheme. When it finds the correct place in the list for the specified value it shifts all the values right by one. The advantage of this is that it is very easy to implement and can work well for small lists. The disadvantage is that it can be costly in moves and potentially needs to check a lot of values before finding the right place in the list. I think this method will be the least efficient.

Task 2:

Task 2 is implementing basically the same thing as task 1, but it starts from the end. This has a chance to cut the number of comparisons in certain cases. It has pretty much the same advantages and disadvantages. I think this will end up being slightly more efficient than task 1's method.

Task 3:

Task 3 leaves gaps in the list to remove the need to shift. This drastically reduces the number of moves needed compared to the methods in task 1 and 2. This method, however, requires the tracking of empty spaces in the list which can be more expensive

Task 4:

```
project$ ./run.sh 4_2_task.cc
After 100 runs:
OrderedList1: Comparisons=47832 Moves=43284
OrderedList2: Comparisons=48248 Moves=42855
OrderedList3: Comparisons=78619 Moves=2588
```

Figure 1: 100 runs with a list size of 30

```
project$ ./run.sh 4_3_task.cc
After 100 runs:
OrderedList1: Comparisons=114936 Moves=107162
OrderedList2: Comparisons=113997 Moves=105961
OrderedList3: Comparisons=213378 Moves=4540
```

Figure 2: 100 runs with a list size of 50

```
project$ ./run.sh 4_4_task.cc
After 100 runs:
OrderedList1: Comparisons=5317 Moves=3893
OrderedList2: Comparisons=5234 Moves=3851
OrderedList3: Comparisons=8142 Moves=701
```

Figure 3: 100 runs with a list size of 10

2. After running this program 100 times, this program lined up well with my expected results from task 3, but I definitely did not expect the changes in the results to be so extreme for the third method. Also, method 2 ended up having more comparisons than method 1 but I guess that is always a possibility.

3. This version of the program had pretty much the same overall results. It is interesting to see how the comparisons and moves almost doubled even though the list of the list did not. Also, method 2 was more efficient in comparisons with the bigger list.

4. In this version of the program methods 1 and 2 were nearly identical. Method three still maintained the same pattern of many more comparisons and less moves.

5. The way the repeated running and reporting of the test cases was implemented began with copying and pasting each unique class into a new file. Then, a for-loop was iterated through 100 times, generating new instances of each list, creating random numbers and inserting them into each list, then removing random numbers from those lists. This process was repeated for each of tasks 4.2, 4.3, and 4.4, resulting in three files that can be executed to immediately receive the required results.

   Another option was to import the classes from the original files they were written in. This method was not chosen because our classes were not constructed to take in the correct parameters to do the testing needed. Thus, it was simpler to make copies of the code and tailor it to our needs for each case.

Concept/Objective Description and Importance

This lab explores optimization of insertion and deletion in an ordered list. It also helped refresh our C++ skill. These concepts are required to succeed in this course because we are constantly working with optimization problems, and we use C++ to test and optimize programs. Additionally, the results show there are many ways to achieve a goal in CS, and each way has its own advantages and disadvantages. It also teaches us how to properly test each of those options, so we can make the most efficient decisions in our code.

In the computer science field, knowing how to optimize basic operations like the ones explored in this lab is essential to one's success. In a job interview, similar concepts may appear, and knowing this topic could be the difference between receiving an offer or not.