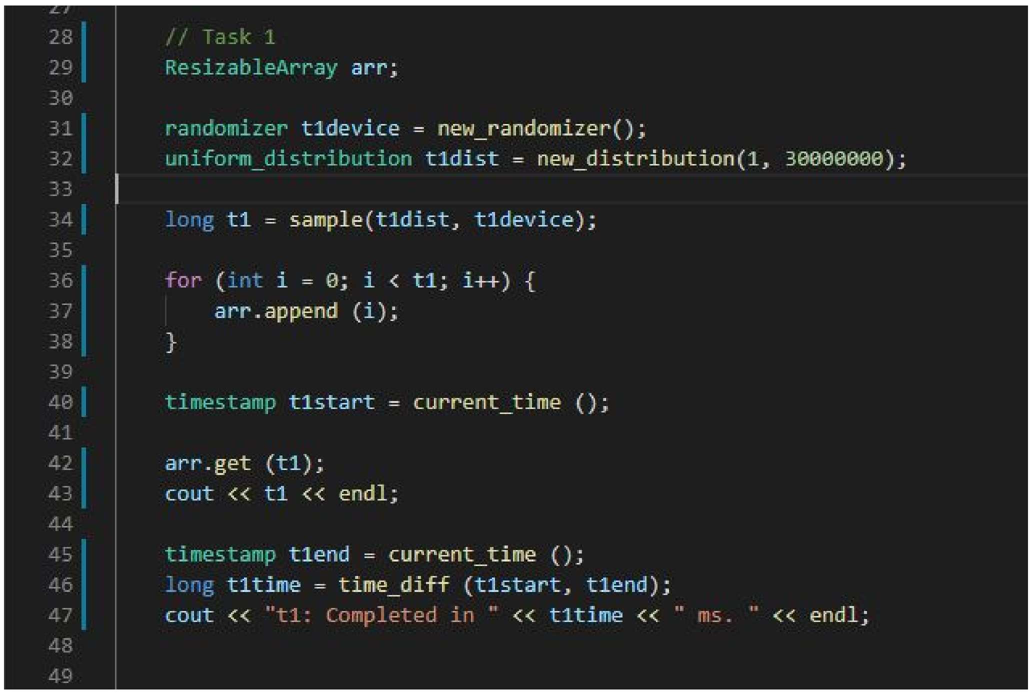
Tony Doan
Professor Kyrilov
CSE030

Lab Report:

For each of the tasks and size adjustments, the code was ran three times. The average of the time it took for each of the three trials was taken and plotted on a graph to show the relationship. The starting size for each task was 5,000,000 and after every three trials ran, the size was increased by 5,000,000 until it reached 30,000,000.

Task 1:

- *Code for Task 1:*

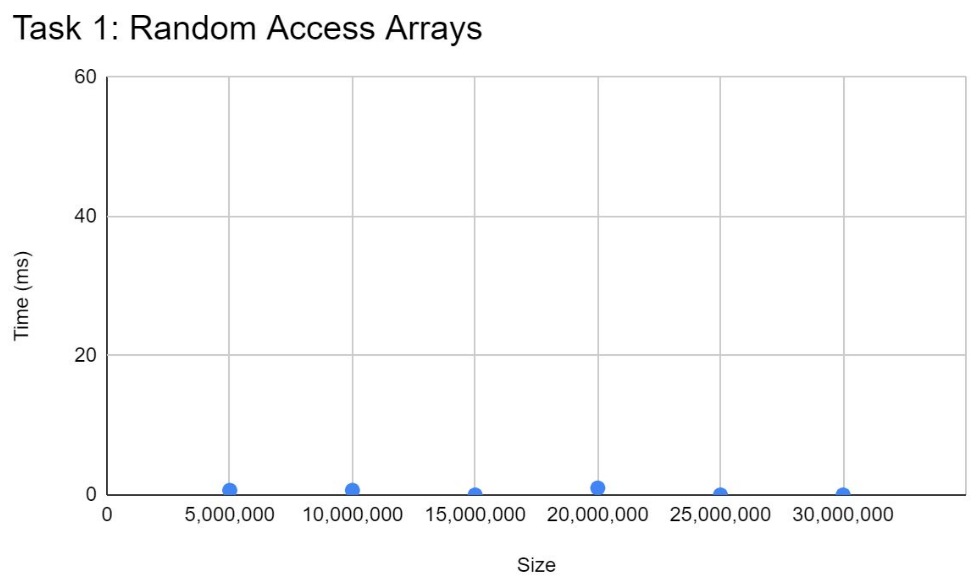


```
// Task 1
ResizableArray arr;

randomizer t1device = new_randomizer();
uniform_distribution t1dist = new_distribution(1, 30000000);

long t1 = sample(t1dist, t1device);

for (int i = 0; i < t1; i++) {
    arr.append (i);
}

timestamp t1start = current_time ();

arr.get (t1);
cout << t1 << endl;

timestamp t1end = current_time ();
long t1time = time_diff (t1start, t1end);
cout << "t1: Completed in " << t1time << " ms. " << endl;
```

For Task 1, I created ResizeableArray arr. Then I created a randomized using “RandomSupport.h” and set “t1” to be the random number that the code was going to pull. The for loop creates an array of the size given and appends “i” each time until the limit is reached. After the for loop finishes, the timer starts, and “arr.get (t1);” grabs a random number and the code after prints it out. Then the timer stops and prints how long the process took.

- *Results for Task 1:*

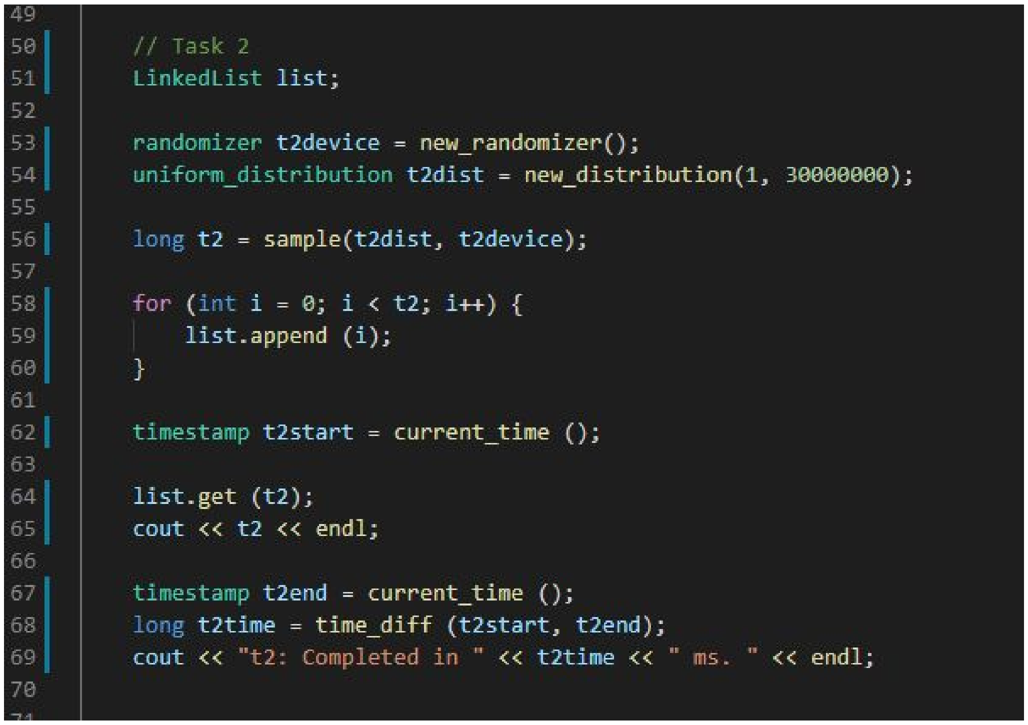
Size	Trial 1	Trial 2	Trial 3	Average
5,000,000	1	0	1	0.6666666667
10,000,000	1	0	1	0.6666666667
15,000,000	0	0	0	0
20,000,000	2	1	0	1
25,000,000	0	0	0	0
30,000,000	0	0	0	0



The majority of the results were 0 ms. The points on the graphs are all below 1 on the y-axis, which shows that the relationship is constant. This is because the code does not have to go through the whole array to pull a random number so it would not take much time.

Task 2:

- *Code for Task 2:*

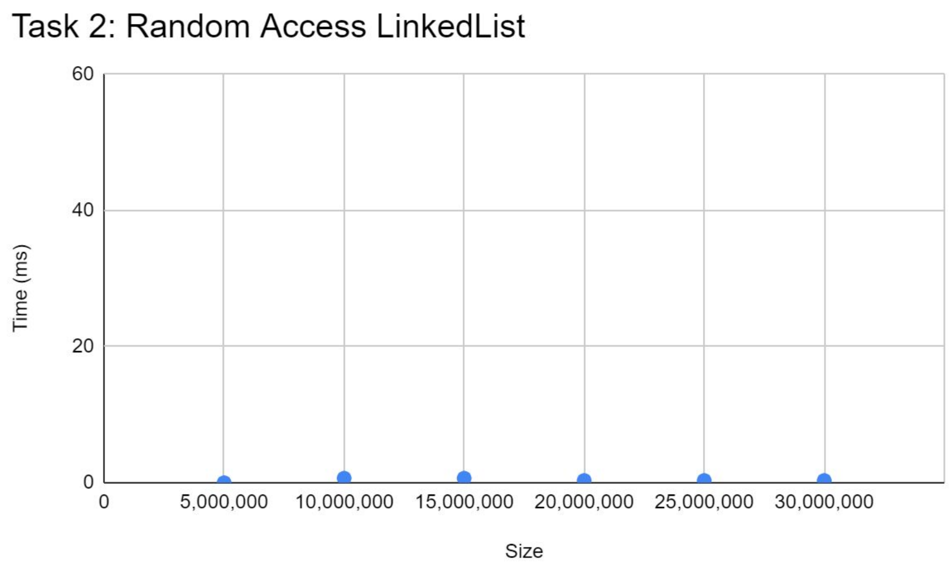


```
// Task 2
LinkedList list;

randomizer t2device = new_randomizer();
uniform_distribution t2dist = new_distribution(1, 30000000);

long t2 = sample(t2dist, t2device);

for (int i = 0; i < t2; i++) {
    list.append (i);
}

timestamp t2start = current_time ();

list.get (t2);
cout << t2 << endl;

timestamp t2end = current_time ();
long t2time = time_diff (t2start, t2end);
cout << "t2: Completed in " << t2time << " ms. " << endl;
```

For Task 2, I basically used the same code as Task 1 but the difference is that I used "LinkedList".

- *Results for Task 2:*

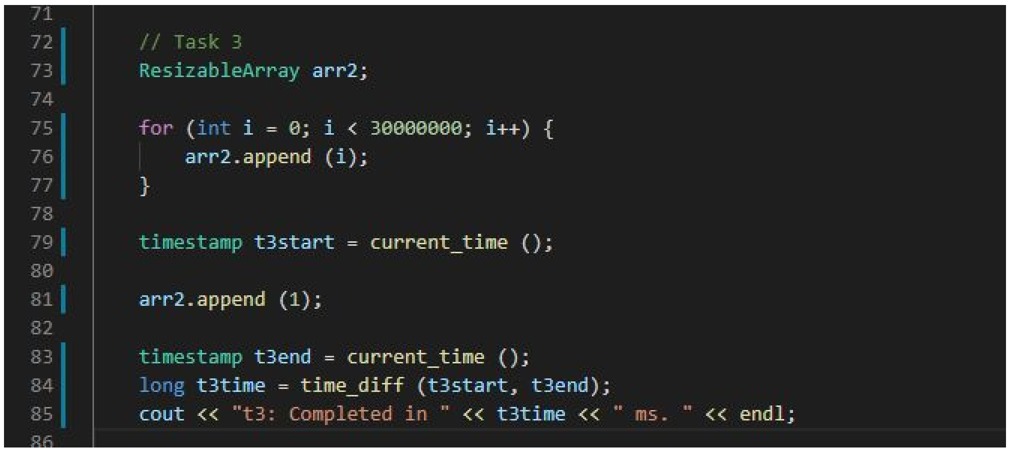
Size	Trial 1	Trial 2	Trial 3	Average
5,000,000	0	0	0	0
10,000,000	1	1	0	0.6666666667
15,000,000	1	1	0	0.6666666667
20,000,000	1	0	0	0.3333333333
25,000,000	0	1	0	0.3333333333
30,000,000	0	0	1	0.3333333333



The results I got for Task 2 were very similar to Task 1. Getting a random number in a LinkedList is constant. However I think that this is wrong, and I think the relationship between getting a random number and the size is linear, because it should take longer to get through a bigger list, therefore time should increase when the size increases.

Task 3:

- *Code for Task 3:*



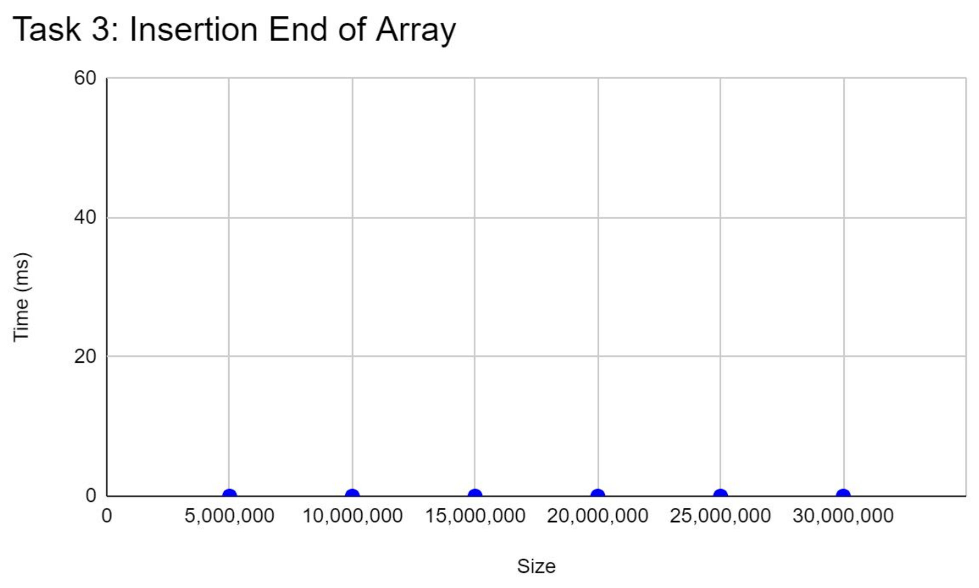
```
    // Task 3
    ResizableArray arr2;

    for (int i = 0; i < 30000000; i++) {
        arr2.append (i);
    }

    timestamp t3start = current_time ();

    arr2.append (1);

    timestamp t3end = current_time ();
    long t3time = time_diff (t3start, t3end);
    cout << "t3: Completed in " << t3time << " ms. " << endl;
```

For Task 3, I created "ResizableArray arr2" and for the other ResizeableArray, I called it arr3 etc. In this, the for loop creates an array of a size I tell it too and appends "i" into each index. Then a timer starts and "arr2.append

(1);" puts the number 1 into the end of the array. After the timer stops and prints out the time it took to run the program.

- *Results for Task 3:*

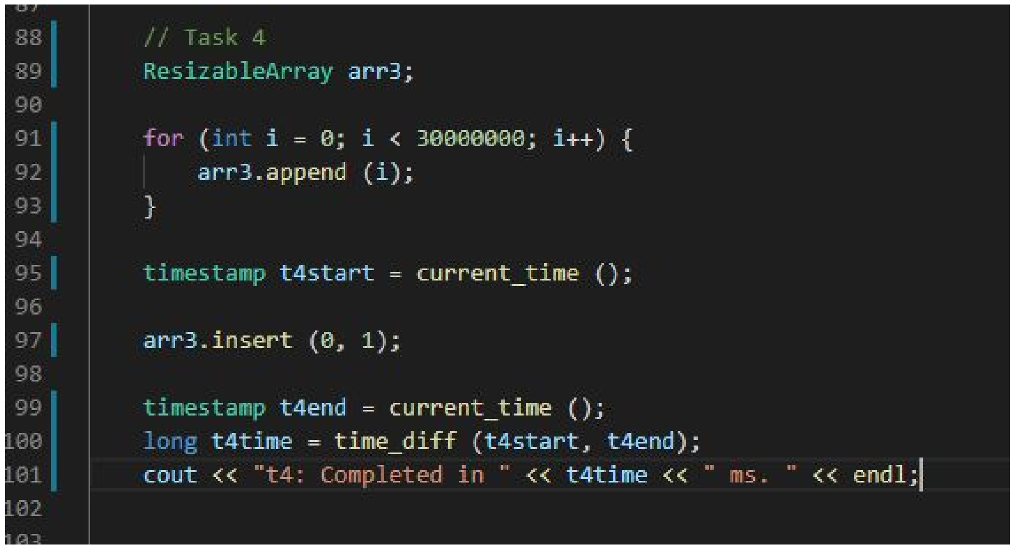
Size	Trial 1	Trial 2	Trial 3	Average
5,000,000	0	0	0	0
10,000,000	0	0	0	0
15,000,000	0	0	0	0
20,000,000	0	0	0	0
25,000,000	0	0	0	0
30,000,000	0	0	0	0



For all the trials for each size in the program, the time it took to append the number 1 into the array was 0 ms. The relationship is constant because it does not change for each size given. This is because the program just goes to the end of the array and inserts the number.

Task 4:

- *Code for Task 4:*

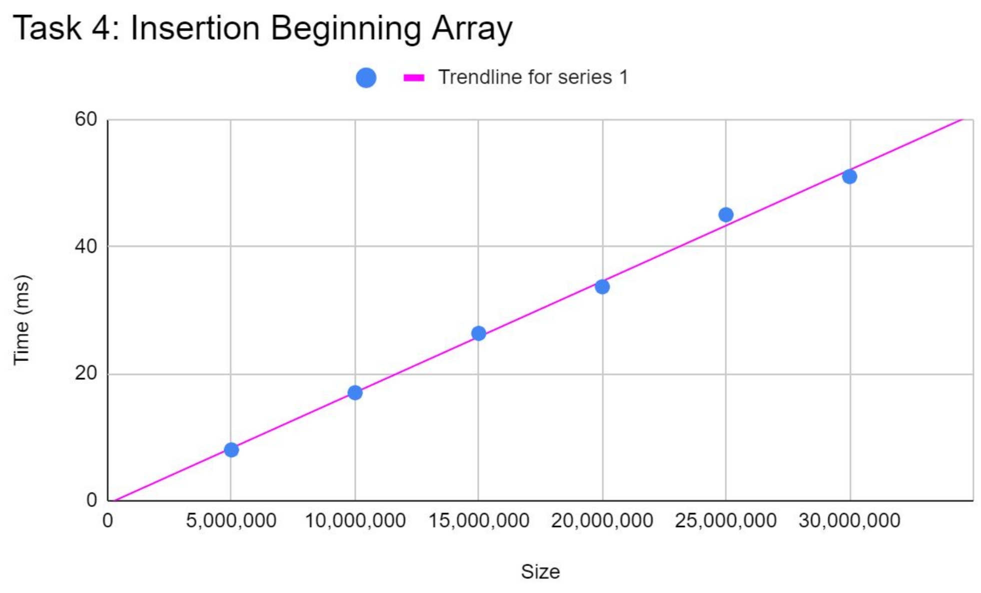


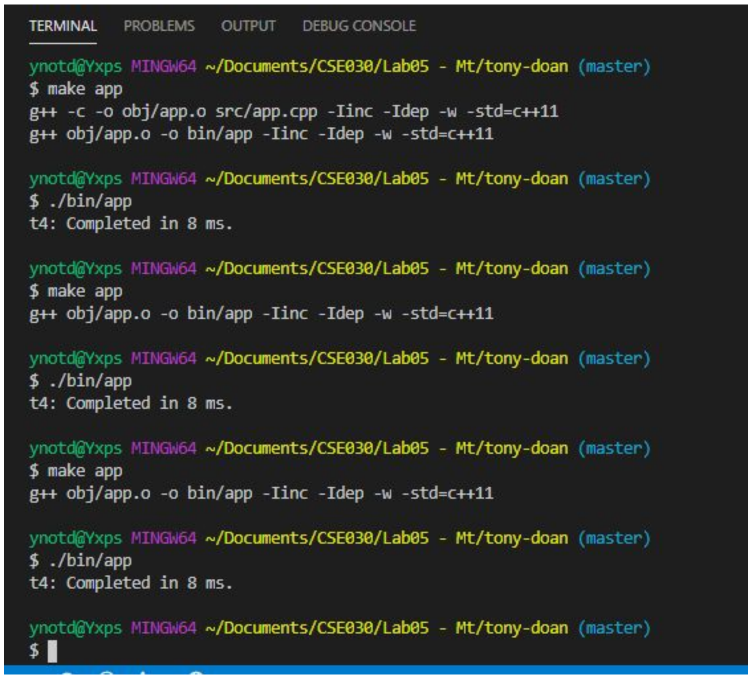
```
// Task 4
ResizableArray arr3;

for (int i = 0; i < 30000000; i++) {
    arr3.append (i);
}

timestamp t4start = current_time ();

arr3.insert (0, 1);

timestamp t4end = current_time ();
long t4time = time_diff (t4start, t4end);
cout << "t4: Completed in " << t4time << " ms. " << endl;
```

Task 4's code is pretty much the same as Task 3's, except the difference is that instead of "arr.append ();" I used "arr.insert (0,1);" to insert the number 1 to the first index.

- *Results for Task 4:*

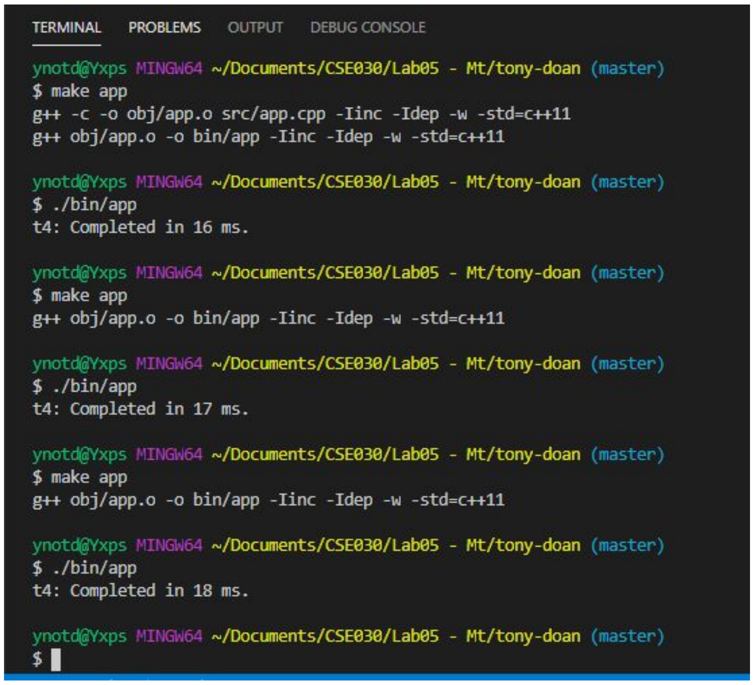
Size	Trial 1	Trial 2	Trial 3	Average
5,000,000	8	8	8	8
10,000,000	16	17	18	17
15,000,000	27	25	27	26.33333333
20,000,000	34	33	34	33.66666667
25,000,000	46	42	47	45
30,000,000	51	51	51	51





```
ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ make app
g++ -c -o obj/app.o src/app.cpp -Iinc -Idep -w -std=c++11
g++ obj/app.o -o bin/app -Iinc -Idep -w -std=c++11

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ ./bin/app
t4: Completed in 8 ms.

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ make app
g++ obj/app.o -o bin/app -Iinc -Idep -w -std=c++11

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ ./bin/app
t4: Completed in 8 ms.

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ make app
g++ obj/app.o -o bin/app -Iinc -Idep -w -std=c++11

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ ./bin/app
t4: Completed in 8 ms.

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ 
```

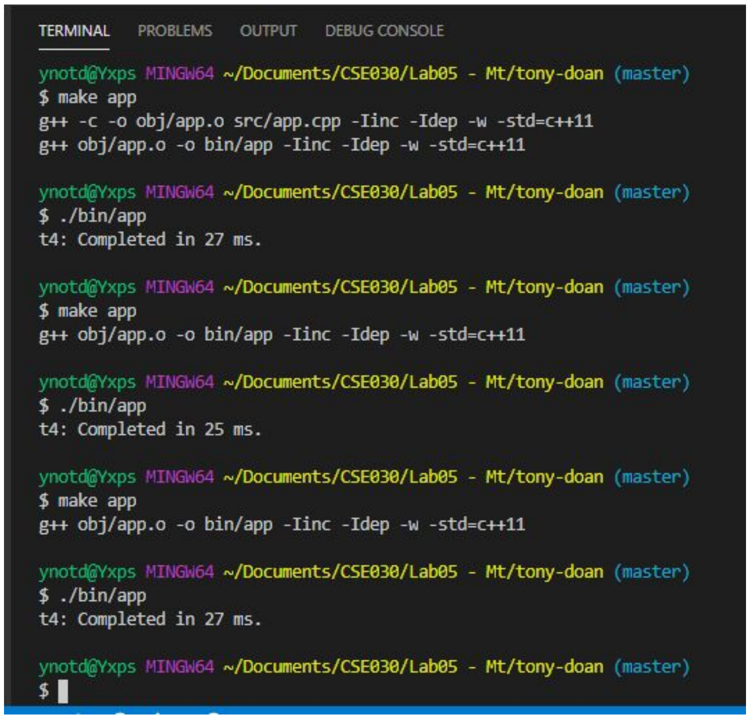
Size: 5,000,000



TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

```
ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ make app
g++ -c -o obj/app.o src/app.cpp -Iinc -Idep -w -std=c++11
g++ obj/app.o -o bin/app -Iinc -Idep -w -std=c++11

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ ./bin/app
t4: Completed in 16 ms.

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ make app
g++ obj/app.o -o bin/app -Iinc -Idep -w -std=c++11

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ ./bin/app
t4: Completed in 17 ms.

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ make app
g++ obj/app.o -o bin/app -Iinc -Idep -w -std=c++11

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ ./bin/app
t4: Completed in 18 ms.

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$
```

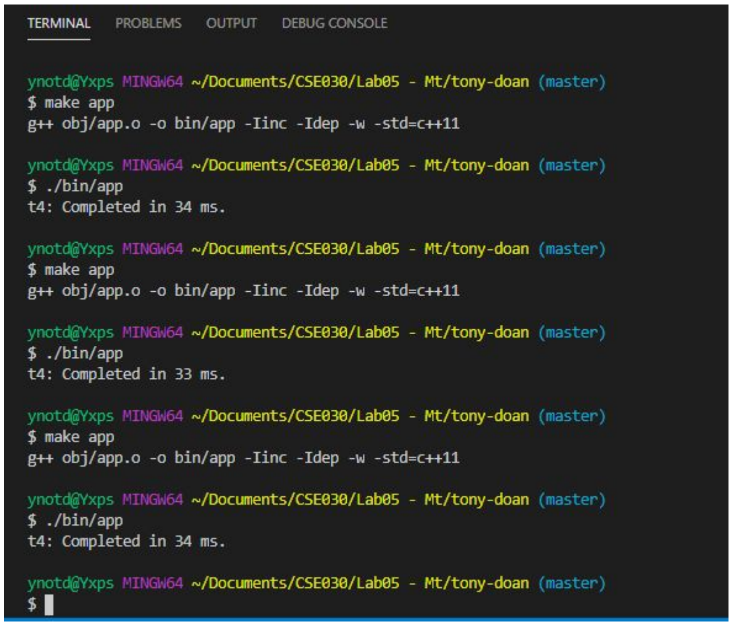
Size: 10,000,000



TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

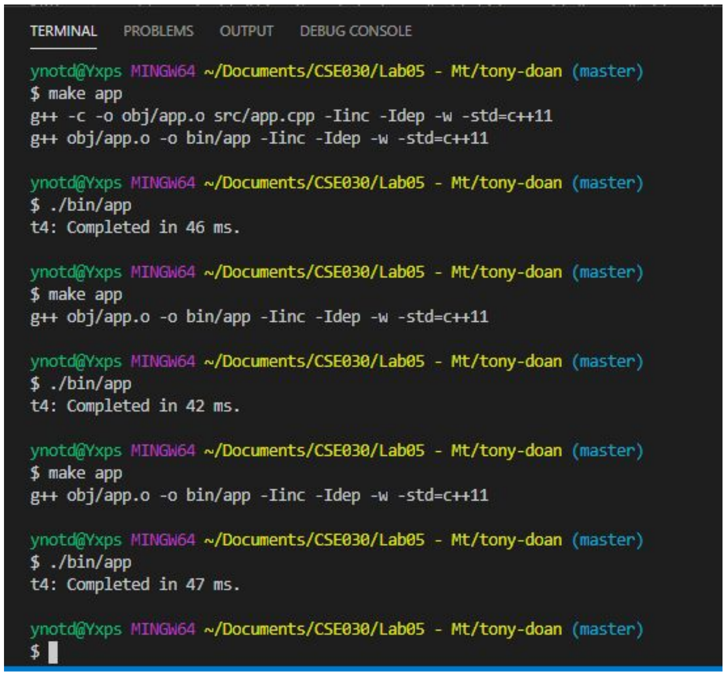
```
ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ make app
g++ -c -o obj/app.o src/app.cpp -Iinc -Idep -w -std=c++11
g++ obj/app.o -o bin/app -Iinc -Idep -w -std=c++11

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ ./bin/app
t4: Completed in 27 ms.

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ make app
g++ obj/app.o -o bin/app -Iinc -Idep -w -std=c++11

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ ./bin/app
t4: Completed in 25 ms.

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ make app
g++ obj/app.o -o bin/app -Iinc -Idep -w -std=c++11

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ ./bin/app
t4: Completed in 27 ms.

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$
```

Size: 15,000,000



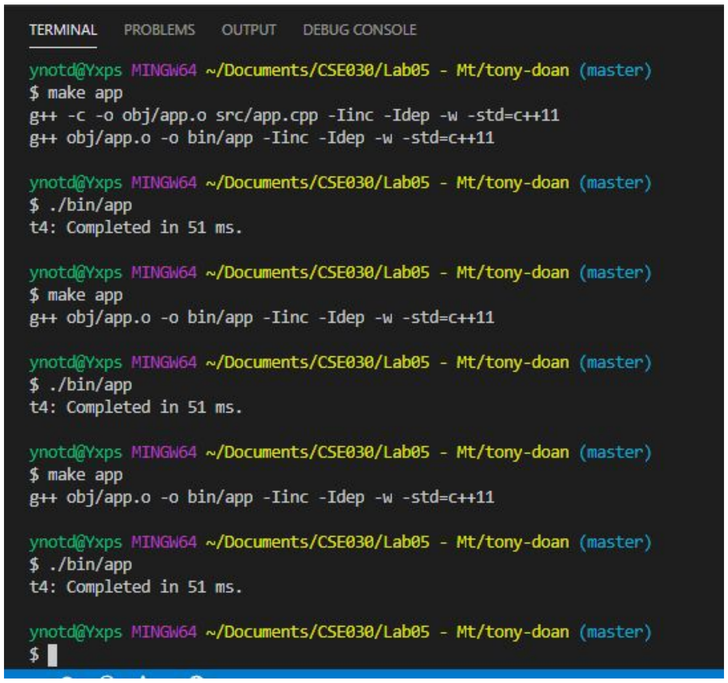
```
ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ make app
g++ obj/app.o -o bin/app -Iinc -Idep -w -std=c++11

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ ./bin/app
t4: Completed in 34 ms.

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ make app
g++ obj/app.o -o bin/app -Iinc -Idep -w -std=c++11

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ ./bin/app
t4: Completed in 33 ms.

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ make app
g++ obj/app.o -o bin/app -Iinc -Idep -w -std=c++11

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ ./bin/app
t4: Completed in 34 ms.

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$
```

Size: 20,000,000



```
ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ make app
g++ -c -o obj/app.o src/app.cpp -Iinc -Idep -w -std=c++11
g++ obj/app.o -o bin/app -Iinc -Idep -w -std=c++11

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ ./bin/app
t4: Completed in 46 ms.

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ make app
g++ obj/app.o -o bin/app -Iinc -Idep -w -std=c++11

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ ./bin/app
t4: Completed in 42 ms.

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ make app
g++ obj/app.o -o bin/app -Iinc -Idep -w -std=c++11

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ ./bin/app
t4: Completed in 47 ms.

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$
```

Size: 25,000,000



```
ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ make app
g++ -c -o obj/app.o src/app.cpp -Iinc -Idep -w -std=c++11
g++ obj/app.o -o bin/app -Iinc -Idep -w -std=c++11

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ ./bin/app
t4: Completed in 51 ms.

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ make app
g++ obj/app.o -o bin/app -Iinc -Idep -w -std=c++11

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ ./bin/app
t4: Completed in 51 ms.

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ make app
g++ obj/app.o -o bin/app -Iinc -Idep -w -std=c++11

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ ./bin/app
t4: Completed in 51 ms.

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ 
```

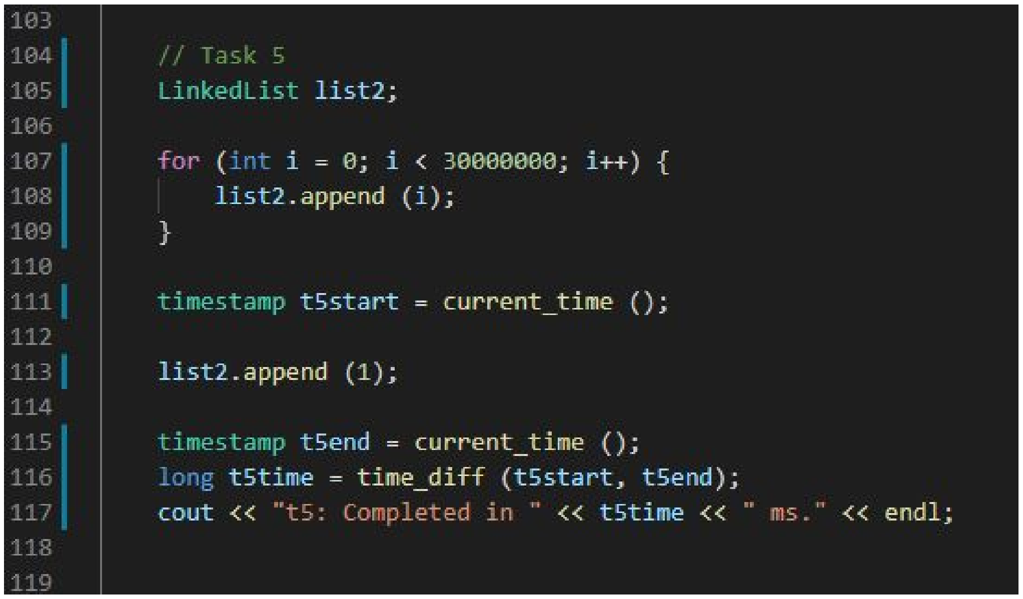
Size: 30,000,000

The results of Task 4's code is different from the others, because it shows a linear relationship. This is because the larger the size of the array, the more time it will take to insert a number in the beginning, since the program has to shift more numbers to make room.

Under the graph are the screenshots of the outputs from the terminal for Task 4. I did not include screenshots for the other Tasks because their outputs were mostly zeros, while this one shows different numbers for each test.

Task 5:

- *Code for Task 5:*

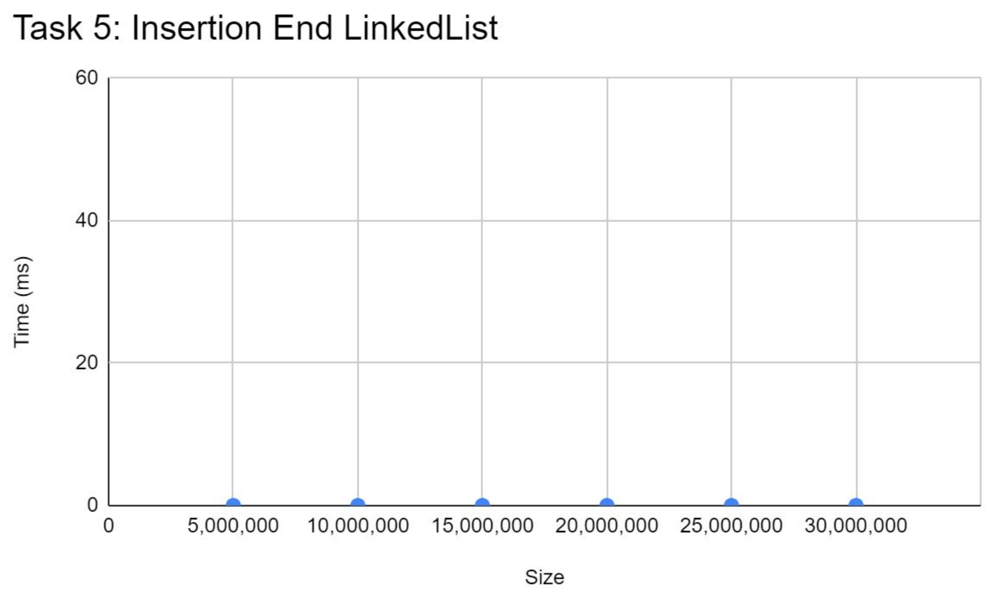


```
    // Task 5
    LinkedList list2;

    for (int i = 0; i < 30000000; i++) {
        list2.append (i);
    }

    timestamp t5start = current_time ();

    list2.append (1);

    timestamp t5end = current_time ();
    long t5time = time_diff (t5start, t5end);
    cout << "t5: Completed in " << t5time << " ms." << endl;
```

For Task 5, I used the same code as for Tasks 3 and 4. However, I used "LinkedList.h" for this task and declared "list2". The for loop creates a linked list with the size I give it and appends "i" to each node. Then a timer starts and the number 1 is appended into the list and the timer stops and prints out the time it took.

- *Results for Task 5:*

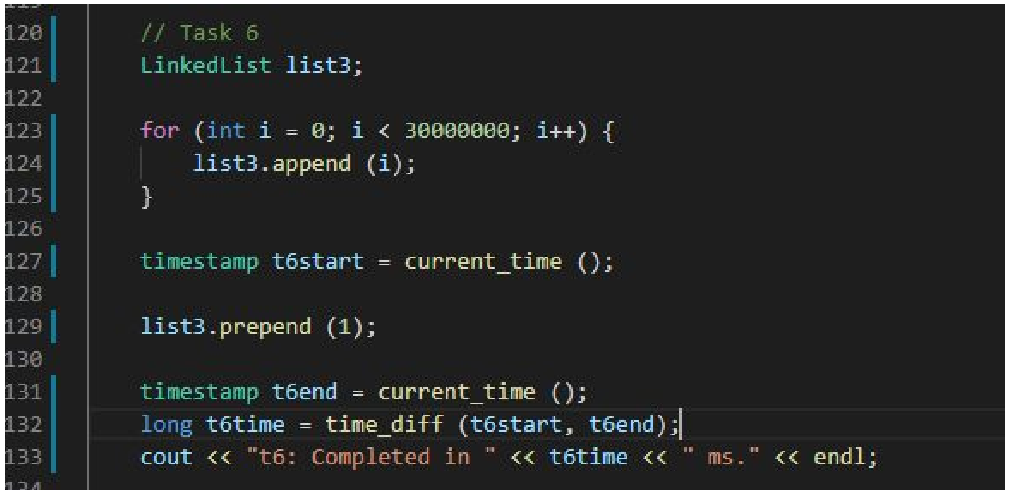
Size	Trial 1	Trial 2	Trial 3	Average
5,000,000	0	0	0	0
10,000,000	0	0	0	0
15,000,000	0	0	0	0
20,000,000	0	0	0	0
25,000,000	0	0	0	0
30,000,000	0	0	0	0



The results for Task 5 are similar to all the other tasks, except 4. The time it took to append the number 1 into a large linked list was 0 ms for each test. The relationship is constant since the output was always 0 ms for each size given. However, I think this could be wrong because it should take a longer time the larger the list because the program has to go one by one through each node until it reaches the end.

<u>Task 6:</u>

- *Code for Task 6:*

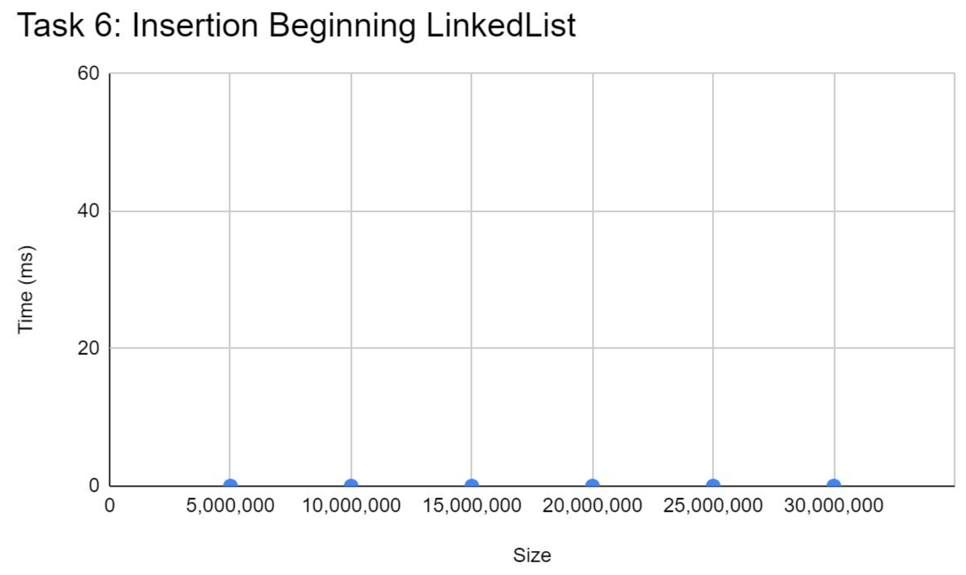


```
// Task 6
LinkedList list3;

for (int i = 0; i < 30000000; i++) {
    list3.append (i);
}

timestamp t6start = current_time ();

list3.prepend (1);

timestamp t6end = current_time ();
long t6time = time_diff (t6start, t6end);
cout << "t6: Completed in " << t6time << " ms." << endl;
```

For Task 6, I used the same code for the previous code for Task 5, but I used "list.prepend (1);". The prepend function will add a number to the front of the list instead of the end.

- *Results for Task 6:*

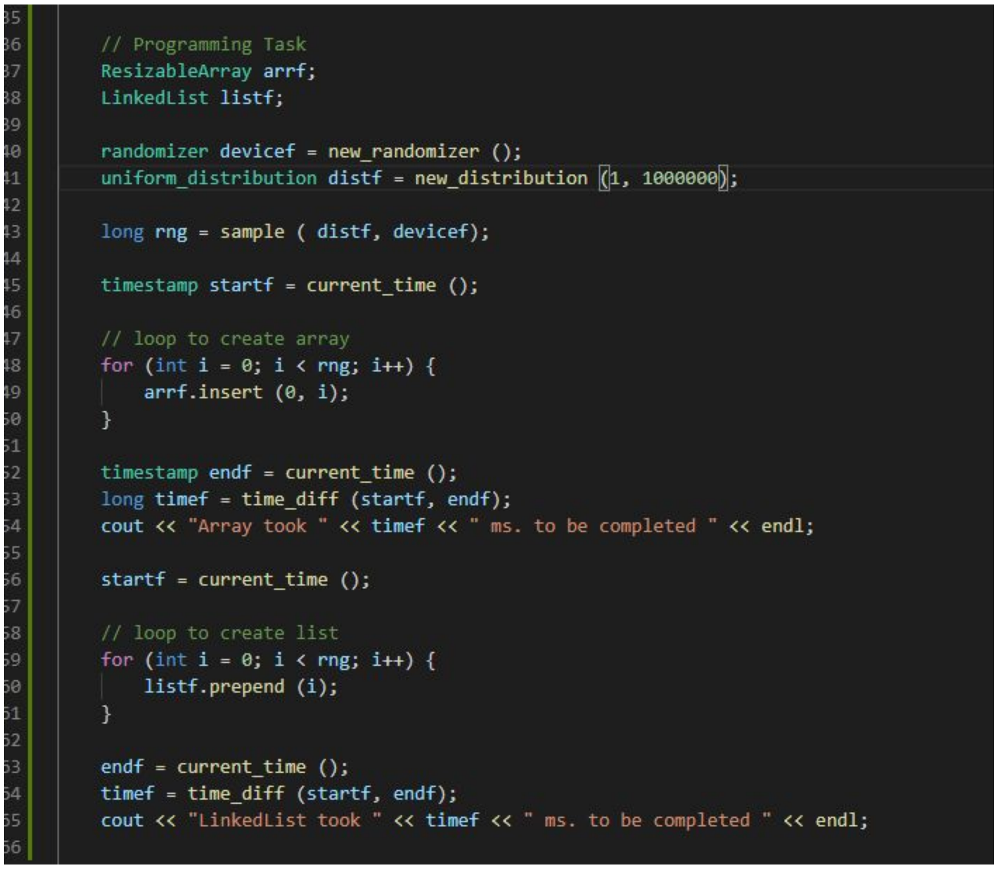
Size	Trial 1	Trial 2	Trial 3	Average
5,000,000	0	0	0	0
10,000,000	0	0	0	0
15,000,000	0	0	0	0
20,000,000	0	0	0	0
25,000,000	0	0	0	0
30,000,000	0	0	0	0



Like the other tasks, the results for this task also had a constant relationship. The time it took to insert a number in the front of a list was 0 ms. This is because the program does not have to go through each node individually since it just has to find the head of the list and create a node in front of the head and set that new node to be the new head.

Programming Task:

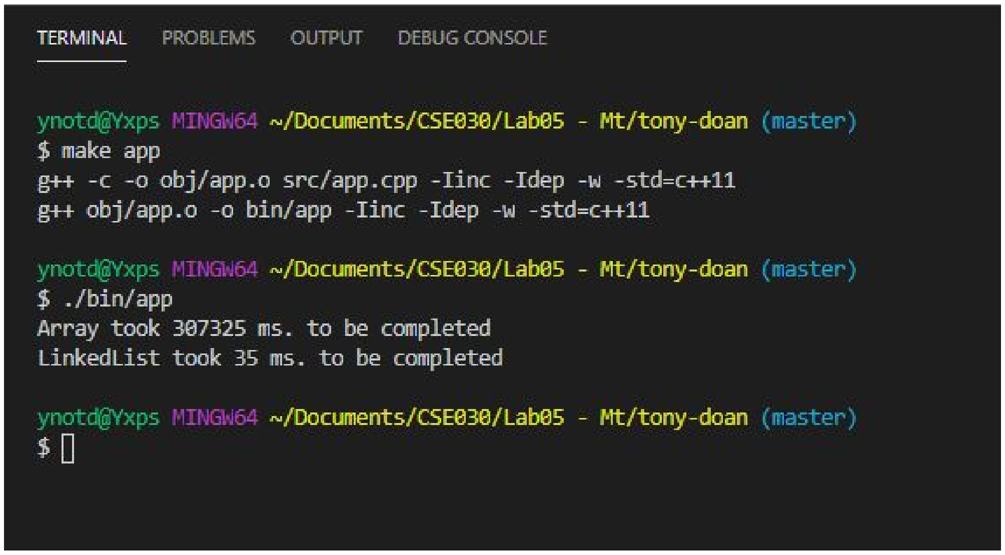
- *Code:*



```
// Programming Task
ResizableArray arrf;
LinkedList listf;

randomizer devicef = new_randomizer ();
uniform_distribution distf = new_distribution (1, 1000000);

long rng = sample ( distf, devicef);

timestamp startf = current_time ();

// loop to create array
for (int i = 0; i < rng; i++) {
    arrf.insert (0, i);
}

timestamp endf = current_time ();
long timef = time_diff (startf, endf);
cout << "Array took " << timef << " ms. to be completed " << endl;

startf = current_time ();

// loop to create list
for (int i = 0; i < rng; i++) {
    listf.prepend (i);
}

endf = current_time ();
timef = time_diff (startf, endf);
cout << "LinkedList took " << timef << " ms. to be completed " << endl;
```

For the final task, I declared “arrf” and “listf”. Then I set a randomizer to get a random number and a start variable for the timer. Once the timer starts, a for loop runs and inserts “i” into index 0 for the large size of the array. Then the timer stops and prints how long it took for the array to insert the numbers. Then the timer starts again and this time, “i” is prepended into a list and the timer times how long that will take and prints out the result.

- *Final result:*



```
ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ make app
g++ -c -o obj/app.o src/app.cpp -Iinc -Idep -w -std=c++11
g++ obj/app.o -o bin/app -Iinc -Idep -w -std=c++11

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ ./bin/app
Array took 307325 ms. to be completed
LinkedList took 35 ms. to be completed

ynotd@Yxps MINGW64 ~/Documents/CSE030/Lab05 - Mt/tony-doan (master)
$ 
```

The array took longer than the list because the array had to insert “i” at index zero and shift all the other numbers individually to make room. The list is more efficient because it is faster and it does not have to shift its nodes and just places “i” as the new head.