Benjamin Poile
11565739
CS 223 - PA2
PA2 Report

A. Problem Statement
    a. We wanted to observe and analyze the performance of a function when run with Vector and Singly-linked list data types. We sought to do this with by writing the solution to the MyJosephus problem in C++.
B. Algorithm Design
    a. For the game mechanics, I decided that the best route to take while writing the function is to initialize the list or vector and advance it's iterator to the index of the player to be removed. To find the index of the player we need to remove, I wrote the following equation:

```
(((numElim * this->M) - 1) - numElim) % this->N
```

    After getting the result of that equation, I skip to that index and remove the node from the list/vector. The size of the data type is adjusted to reflect the new length. My timing function takes in a function as a parameter, allowing me to call it within another function. The reason I do this is to measure the runtime of functions in a clear, modular way. When the function is passed in, a timer is started (clock()) and the parameter function is run. When the function completes, the output data of the function is written to a struct, along with the runtime. The timing function returns this struct.
    I used the standard library for both the linked list code and the vector code, per instruction.
C. Experimental Setup
    a. Machine Specs: MacBook (Retina, 12-inch, 2017); 1.4 GHz Intel i7; 16GB RAM
    b. OS and environment used when testing: MacOS Mojave 10.14; C++ 11
    `gcc *.cpp *.h -lstdc++ -std=c++11 -w -o out ARGN ARGM`
D. Experimental Results and discussion.
    a. The vector implementation here proved to be the faster route, in my tests it consistently outperformed the other implementation. This won't always be the case, though. This project dealt with small set of data, so scalability isn't a problem, but if this experiment entailed the handling of millions of data points, the performance difference between the two is much more obvious.
    There are advantages and disadvantages of each data type, but generally speaking, a vector is the better of the two for these sorts of purposes. The biggest difference between the two data types is their scalability. Linked Lists are very good at insertion and deletion, but struggle when it comes to referencing data in the middle of the object, as it must traverse the list to access. Vectors perform at scale much better than Linked Lists, as their runtime complexities for inserting, especially towards the back and middle of a list is much faster.