Name: Yatin Singla

Date: 5th February 2018

# Programming Assignment 2

## Report

1. The goal of this programming assignment was to identify the performance and implementation tradeoffs between the linked list and array ADTs by implementing the Josephus problem game. Where N number of people sit in a circle and play the game. The game ends when only one person is left. People have to pass a potato certain number of times before the person holding the potato is eliminated.
2. The main idea behind my implementation of the game was to use a maximum number of inbuilt functions that come along with STL lists and vectors. The game is simulated by initializing the class with values of N and M which help set other two data members passes and size. Passes keep track of the number of potato passes left before somebody is eliminated. Circular linked list and array is implemented with the help of iterators. The program keeps track of the address to which iterator points to and if iterator == object.end() then the program sets the iterator back to beginning as follows: iterator = object.begin(). Henceforth the idea of circular linked list and array are implemented.

A player is eliminated using the function erase. Erase function uses the STL function erase for deleting a certain element in lists and vectors with the help of iterators.

1. Machine Specification:- Intel I7 6700HQ processor with a clock speed of 2.6 GHz (8 CPU). Operating System: Windows 10 (64-bit). RAM: 16 GBs.

I repeated each experiment at least 3 times before reporting the final timing statistics.

Windows machine with Visual Studio were used to test the program.

Discussion and Report:

After analyzing the data, it pointed out that in most case the total game run time was quicker for the vector implementation of the game, but not by much for the tested data values. The data also pointed out the game runtime for vector implementation was quicker irrespective of N and M values in all cases.

However, the data pointed otherwise when we compared average elimination time for both implementations. In some cases, list implementation was faster whereas in other cases the vector implementation was faster. List implementation outperformed vector implementation at the removal of player as the number of players increased. This outcome is theoretically backed up as when we remove an element from an array all the elements following that element are rolled over by 1, which accounts for the lost time. Overall, in three of the four graph comparisons, my vector structure is faster than my list structure. Trends that I noticed when collecting data points is that time difference is minimal when comparing N-values of 4 to 16, and M-Values of 4 to 16. This could be because my computer compiles my code so quickly that a difference of 12 Person objects and a difference of 12 turns until elimination doesn’t make a notable difference in terms of time spent processing. Another worthy note is that both the data structures didn’t have a notable time difference where we could blatantly say if one data structure was better than the other over all the test cases that we ran. My data shows that for almost all the test cases the time complexity of the program showed a linear trend with a few exceptional data set here and there.