Group 7

CSCI 313

HW 1-- Question 1: Vector vs. List

First, header files that allow for vectors, lists, use of randomizing variables, and measuring execution times are included. A template is created that allows us to measure the amount of time a function takes to execute (TimeFunc) by subtracting the time taken at the beginning of the function from the time taken at the end of it. TimeFunc is used multiple times throughout int main() to compare the lengths of time between the vector and the list.

VectorofRandomNumbers takes in a vector and its size, and uses a for loop to push back random numbers into the vector. It does this until the vector is full. ListofRandomNumbers does the exact same thing, but instead pushes back random numbers into a list, not a vector.

GetRandomString is used for creating and returning random strings. A length is given to determine the length of the random string, then the string is created with a character array. A switch-case statement is used, and 'flag' is filled with a random number from 0 to 2. It then determines based on flag whether to input a character into the array that is a capital letter, lowercase letter, or a number from 0 to 9. These three choices are based on ascii. This loops until the last element is reached, and then the loop finishes. The last element is '\0', which marks the end of the string. The character array is passed into an actual string str, which is then returned to wherever the function is called.

VectorOfRandomStrings takes in a vector and a size and fills the vector with random strings using getRandomString and a for loop. A random number from 1 to 50 is generated to pass into getRandomString as the string's length. Then it pushes this string into the vector with v.push_back(str), then it loops until the vector is full, which is when the loop will have reached

size - 1. ListofRandomString does the same thing as VectorofRandomStrings, but it takes in and fills a list with strings instead of a vector, and it also uses getRandomString.

VectorOfRandomStrings_Move also does the same thing as VectorofRandomStrings, except it uses move semantics to push the random strings into the vector. Instead of giving a copy of the random string to v.pushback, it directly moves the contents of str into the vector. ListOfRandomString_Move does the same thing but pushes random strings into a list, not a vector.

In main, the first line of code grabs a random seed with srand so that rand() does not generate the same 'random' value when it is called. A constant size of 10000 is also established to be used for the vectors and lists. The rest of main is separated into 3 parts for each part of the question. In all three parts, an integer vector and integer list are declared. Then functions for each part for filling vectors and lists are executed, and their execution times are measured with TimeFunc, then the times are outputted to console and compared.

For part a, a vector and a list are filled with 10,000 random integers using VectorOfRandomNumbers and ListOfRandomNumbers. With TimeFunc, their execution times are measured, and then outputted to console. An if statement is used to state which of the two is faster, and after running the program multiple times, it seems that filling a vector with random integers is faster than filling a list with random integers.

For part b, the vector and list are filled with 10,000 random strings using VectorOfRandomStrings and ListOfRandomString. Just like in part a, their times are stated, and then compared, but unlike part a, filling a vector with random strings is slower than filling a list with random strings. Part c does the same thing as part b, but move semantics is used to fill the vector and the list with VectorOfRandomStrings_Move and ListOfRandomString_Move. The

results show that filling a vector with random strings using move semantics is slower than filling a list with random strings using move semantics, just like in part c.

It seems that, in the case of random numbers, filling a vector is faster than filling a list. But when it comes to random strings, filling a list is faster than filling a vector, with and without move semantics.