Name: Tian Zheng

Student Number: 155 394 174

Email: [tzheng26@myseneca.ca](mailto:tzheng26@myseneca.ca)

1. The reason for using the vector container rather than any other available in the STL.

Because the elements in the container are in sequential order. In addition, the size is unknown in advance, so I cannot use std::array. Plus, there will not be any need adding or removing elements frequently at the front or back, or in the middle of the sequence, so I do not need to use std::deque or std::list. To sum up, std::vector is by default the good top choice as an alternative to built-in arrays.

2. List the STL template functions that you used in your solution. std::accumulate(ys.begin(), ys.end(), (T)0);

std::sort(v.begin(), v.end());

std::inner\_product(tempx.begin(), tempx.end(), tempy.begin(), (T)0);

3. Identify where you used lambda expressions.

In void displayStatistics(std::ostream& os) const:

T s, i;

std::vector<T> tempx = xs;

std::vector<T> tempy = ys;

auto f = [=](T& slope, T& ysIntercept) {

T totalXY = std::inner\_product(tempx.begin(), tempx.end(), tempy.begin(), (T)0);

T totalX = std::accumulate(tempx.begin(), tempx.end(), (T)0);

T totalY = std::accumulate(tempy.begin(), tempy.end(), (T)0);

T totalXX = std::inner\_product(tempx.begin(), tempx.end(), tempx.begin(), (T)0);

T no = tempy.size();

slope = (no \* totalXY - totalX \* totalY) / (no \* totalXX - totalX \* totalX);

ysIntercept = (totalY - slope \* totalX) / no;

};

f(s, i);

4. Comment on the ease of programming associated with the STL.

(1) There is no need to know the size of the container for looping through all the elements.

(2) The template functions provide lots of convenience for calculation.

(3) Values of elements in a container can be directly assigned to anther temporary contaniner.

(4) Adding or dropping elements can be done with straightforward functions.