Explain in your own words what you have learned in completing this workshop. Include in your explanation but do not limit it to the following points (40%):

• The reason for specializing the **sum()** member function.

• The reason for specializing the initial value for a summation.

• The reason for defining the class variable outside the class definition.

To avoid deductions, refer to code in your solution as an example of your implementation of the concepts that you describe.

Include all corrections to the Quiz you have received (30%).

1. I have learned by specializing the sum() member function, sum() can perform appropriate operations on V object based on what type V is. For example, in the class SummableLVPair the function sum() is defined to use addition operation:

V sum(const L& label, const V& sum) const {

V result = LVPair<L,V>::getValue()+sum;

return result; }

For data type like int, double, float, there is no problem with addition operation. However, when it comes to std::string type, a simple addition operation is not enough. Although in std::string class, the operator + is overloaded and so usable, the formatting would be messy if we just simply connect two std::string objects with a “+”. Moreover, then the target of the summation is char type data, the operator “+” cannot be used. Thus, specializing sum() is necessary to deal with summation of std::string:

std::string result = value + " " + LVPair<std::string, std::string>::getValue();

2. Another specialization is specializing the initial value for a summation. Because it is unsure that what data type the V will be, and it would be wrong to initialize a char variable as 0 or to initialize an int variable as “ ”. Specializing the initial value for difference data types ensures to cover all the possible cases and avoids exceptions:

template<>

int SummableLVPair<std::string, int>::init = 0;

template<>

std::string SummableLVPair<std::string, std::string>::init = " ";

3. I have also learned defining class variable must be outside of the class definition. Because class variable is the static variable, which belongs to the whole class. There is only one of a class variable, no matter how many objects there are. If it is defined inside class, then every creation of object to that class would affect the value of class variable.

4. I have also learned some features of std::string. std::string is more convenient than C-style null-terminated string because it has overloaded operators ==, +, etc. Comparing 2 C-style strings by using == would generate error, but doing so on 2 objects to std::string would work:

if (label == temp[i].getLabel()) {

//statements inside

}

5. I have also learned that a class function is he interface for class variable. It can only access to class variables, and cannot access to other data members.

I have also learned the syntax for specializing a member function from class template. After searching for a lot of examples I finally realized that an identifier of a class template must be followed by either the general typenames or the specific types, which enclosed in < >. The complier cannot identify the class template if the type names are missing.

Correction for quiz:

Question:

char  name[] = "Jane Doe";

char\* surname = NULL;

surname = &name[5];

surname[-1] = '.';

cout << surname << endl;

Answer: Doe