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

The purpose of template specializing is to create the exception of the template definition. By default, the sum() member function will accumulate the two values. However, when it receives two string parameters, we want to insert a single space between two values. As a result, we need to use template specializing to fulfill the goal.

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

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

}

By default, it will calculate the values.

template<>

std::string SummableLVPair<std::string, std::string>::sum(const std::string& label, const std::string& value) const {

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

}

With template specializing, we can concatenate two strings.

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

The reason is similar to the first question. We use specializing to create the exception to match the particular type. Because the logic of calculating string and numbers are different, we need to assign 0 for int and empty space for string. Without specializing the initial value, the result might be different from our expectation.

template<>

int SummableLVPair< std::string, int >::value{ 0 };

in the case of int, initialize the value as numbers.

template<>

std::string SummableLVPair<std::string, std::string>::value{ "" };

in the case of string, initialize the value as empty string.

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

Class variable(static member) is not the part of the individual class object, it belongs to all the objects created by that class. As a result, we need to initialize it outside the class. In addition, the static member variable is usually placed directly underneath the class.

template <typename L, typename V>

size\_t SummableLVPair<L, V>::swidth = 0u;