The difference between the implementations of a functor and a lambda expression.

Functor:

Functor is an object which can be treated as the function. The power of functor is that it can contain the state. In the functor, we will overload the operator () to let the user call it like the normal function.

Ex:

float operator()(float price) {

return price \* (1 + taxRate);

}

lambda expression:

Lambda express is also known as the closures, which means the function in another function. The power of the lambda function is that it can capture the outer( non-local variable) by using [=] or [&].

Ex:

[=](float grade)->std::string {

if (grade >= 90) {

return "A+";}

A list of the exceptions that you have reported and the cases where you terminated execution normally.

1. in KVList.h operator[]

const T& operator[](size\_t i) const {

if (i < this->length)

{

returnValue = list[i];

}

else {

throw "The index passed to the subscripting operator is out - of - bounds.";

}

It will throw the exception if the index is greater then length, which means out of bounds.

1. In KVList.h one argument constructor

KVList(int n) :KVList() {

if (n > 0)

{

this->length = n;

list = new T[n];

}

else

{

throw n;

}

}

If receive the parameter that is not the positive value, which means it can not set the size of the dynamic memory allocation, it will throw the exception.

1. In w5.cpp try to transfer the grade to character.

gradeList.display(std::cout, [=](float grade)->std::string {

…else if (grade >= 0) {

return "F ";

}

else {

throw "Wrong Grade";

}

});

If the grade received is less than 0, which is not a proper grade, it will throw the exception.

1. In w5.cpp

Every exception will be handled in the try{} catch{} and display the proper error message.

catch (const char\* errMsg) {

std::cout << "\nERROR: " << errMsg << std::endl;

exit(2);

}

catch (int& value) {

std::cout << "\nERROR: " << "The number of elements received in the constructor is not positive-valued. Received: " << value << std::endl;

exit(3);

}

catch (std::exception& ex) {

std::cout << "\nERROR: Something Else Wrong" << ex.what()<<std::endl;

exit(4);

}

Quiz reflection:

Full mark in last quiz.