Lab Assignment: Exception and Template

In this assignment, we are going to solve a program by using the knowledge of Exception, Template, friend function, etc. Please study the following problem description and prepare your solution for the question.

To better understand the Storage Container problem, review the section 16.4 Class Templates and the example SimpleVector.

The member functions have been defined inline. Modify this so that the member functions are defined outside the class.

Please submit the solution on the Blackboard website by Thursday, 15 April 2020. Thank you.

Question : The Storage Container Class

In this program, we are going to implement the Storage template class. The object of this class can act like an array variable. Here is the partial and incomplete declaration of the class:

```
class Storage // this class declaration is incomplete
{
private:
    T *array;
    int size;

public:
    // 1: basic constructor
    Storage(int length)
    { // allocate memory and initialize

    }

    // 2: copy constructor
    Storage(const Storage &elem)
    {
        // size = elem.array_size();
        // allocate memory
        // copy individual elements from the array
    }


    ~Storage()
    {
        // release all the allocated memory
        delete[] array;
        // cout << "\nDeleting all the array elements ...";
    }

    // returns the size of the array
    int array_size() const
```

```
    {
        return size;
    }

    // 3: overloads the square operator so that the object can be used
    // like an array
    // Review section 16.4 in the C++ Book
    /* operator[]() // this declaration and definition are incomplete
    {
        // check the index and throw exception if necessary
        if (index >= size)
            // throw exception

        return array[index];
    } */


    void display() const // display all the elements of the array
    {
        cout << "\n";
        // manually displaying things
        for (int i = 0; i < size; i++)
        {

            cout << array[i];
            if (i == size - 1)
                continue;
            cout << ", ";
        }
    }

    // 4: friend function declaration for ostream << operator
    // to display all the elements by using cout
    template <class CT>
    friend ostream &operator<<(std::ostream &os, Storage<CT> elem);


    // 5: declare the friend template function maximum in the following
    // friend maximum(elem); // declaration is incomplete

    // 6: declare the friend template function searchElement in the following
    // friend searchElement(searchVal, elem); // declaration is incomplete
};
```

1. The class uses pointer variable in the back end. Firstly, the class has a constructor that takes one integer parameter, named size. By using the size variable it will allocate that many numbers of element for the array.

2. Secondly, we need to define a copy constructor for the class. Please remember, the member variable of this class is a pointer. Therefore, direct member wise copy will not work.

3. Third, since, the object of the Storage class will act like an array, we need to overload the square operator in this class. By using the overloaded function, we would be able to write the following code:

```
 Storage<string> myStorage(3); // array of three elements

   myStorage[0] = "Truman";
   myStorage[1] = "Washington";
   myStorage[2] = "Ottawa";
```

In addition, the class has three friend functions. Those friend functions need to be declared inside the Storage class. However, they are not part of the Storage class and their implementation should be done outside the class. These friend functions are also template functions.

4. The first friend template function will overloads the ostream << operator for the ostream objects (so that we can display the Storage object by using cout). By using the implementation of this we want to be able to do the following:

```
cout<<myStorage << endl;
```

and it will display the following:
[Truman, Washington, Ottawa]

5. The second friend template function will determine the max value out of all the elements of the storage class. For example,

```
string maxElement = maximum(myStorage);
cout<<maxElement << endl;
```

It will display: Washington

6. The third friend template function will search for a specific value in the storage object. For example,

```
string searchValue = "Ottawa";
bool returnFlag = searchElement(searchValue, myStorage);
```

The returnFlag will be true, since the searchValue exists in the myStorage object that we have defined previously.

Exception Handling

7. Lastly, we need to define a custom Exception class named StorageException that inherits the runtime_error class in our program. Exception should be generated when the following occurs in your program

a) You are trying to access an element from the Storage object that does not exist. For example, myStorage[10] does not exist, the index value is larger than the size of the array. Hence, in your implementation you should throw an exception when the index value is greater than equal to the size value of the Storage object. Use the implemented StorageException class to throw your exception and catch them in your program.
b) In the friend function maximum and serchElement, if the Storage object size is 0 then you should throw the StorageException with appropriate message.
c) Handle the thrown exceptions in the main function. Include a default catch handler as well.

We have provided a driver program and partial skeleton of the class. Please study them to understand how the class and the friend function will work together in this program. In addition, use sufficient comments to explain the logic of the function code throughout the program.

You need to provide screenshots of the program that demonstrates the features of 4,5,6, and 7.

Please note that, the Storage class should at least work for the following data types, int, long, unsigned, float, double, string, char.

If your program has warnings related to pointer then there will be deduction. Please pay extra attention in the implementation of the copy constructor of the Storage class. Please let me know if you have any questions. Thank you.

Assignment rubrics | Total points: 30

1. The program defines constructor (allocates memory and initializes the instance variables): **2** points
2. The program implements the copy constructor (allocates memory and copies the elements from the argument into the instance variable of the class): **2** points
3. The program overloads the [] operator: **3** points
4. The program is able to implement and demonstrate the display functionality by overloading cout: **2** points
5. The program is able to implement and demonstrate the maximum functionality from the collection: **3** points
6. The program is able to implement demonstrate the search functionality implemented by the friend template function: **3** points
7. The program implements the Exception class (**2** points), uses the exception class to throw exceptions (**2** points), and handle exception in the main function (**2** point)
8. Sufficient comments were added to explain the logic and document the code in 2, 3, 4, 5, 6, and 7. (**5** points)
9. Screenshots provided that shows the features of 4, 5, 6, and 7: (**4** points)