Dry Section:

1) The necessary requirements for the generic type T in the SortedList structure are as follows:

1. **Copy Constructor:** The type T must support creating a copy of an object, as elements may be copied during operations such as list additions or duplications.
   Example:
   At the apply and insert functions:
   result.insert(operation(t));
   Here, insert receives an operation that takes a T object and returns another T object. Therefore, T must have a copy constructor.
2. **Destructor:**
   The type T must support proper destruction, as elements are dynamically allocated and need to be correctly deleted when no longer in use.
   Example:
   head = new Node(sortedList.head->element, nullptr);
3. **Operator >:**
   The type T must implement the > operator to maintain the order of elements in the sorted list.
   Example:
   while (tmp->next != nullptr && tmp->next->element > element) {…}

2) If we implement a const-iterator for the SortedList, it will allow access to modify the elements it points to. This could potentially violate the sorting order of the structure.

3) The required functionality can be implemented using the filter function as follows:

```
List.filter([x](const int& num) -> bool { return num % x == 0; });
```

In this case, list is a list of integers, and x is the number the student wants to filter the list by (i.e., identifying the numbers in the list that are divisible by x).

We need to process the input 0 by throwing an exception.