National Research University Higher School of Economics
Faculty of Computer Science
Bachelor's Program in Data Science and Business Analytics (DSBA)

Introduction to Programming, workshop 27.

Use the provided template that reads the Titanic data set and fills a vector with **`Passenger`** objects.

GitHub: https://github.com/dsba-z/week14cpp2021

Replit: https://replit.com/@l8doku/Workshop27STL

You should apply all functions to collections of **Passenger** objects in all tasks. If you need to compare several objects, you compare passengers. If you need to add numbers together, you add passengers.

Start from task 3

Task 1. Applying functions to collections. (previous workshops)

Round all Fare values to integer numbers using **`std::for_each`** or **`std::transform`**.

Task 2. Filtering. (previous workshops)

1.

Output all different values for fields Parch and Sibsp. Use **`std::unique`** and a custom comparison function.

2.

Create a vector that has only passengers with Fare less than 10 who embarked at port S. Use **`std::remove_if`**. https://en.wikipedia.org/wiki/Erase%E2%80%93remove_idiom

Create a vector that has only passengers with surnames starting with a letter from A to L. Use **`std::copy_if`** and a custom function.

Passengers should be sorted by their id.

3.

Create a vector that has only passengers with Fare less than 10 who embarked at port S, but does not contain any passengers with names starting with letters A to L. Use the previous two vectors and the function **`std::set_difference`**.

4 (optional).

Round down and sum up all Fares for passengers of PClass 3 with no siblings or spouses.

Task 3. Other containters.

Create a set of **`Passenger`** objects using **`PassengerComparator`**. Repeat task 2 using a set instead of a vector.

Ideally, all your code should be exactly the same, with the only difference being the type of the object you use. This isn't strictly possible because **`std::remove_if`** doesn't work with sets. Change it to **`std::copy_if`** for both functions.

The problem of repeated code can be solved using templates later.

Task 4. Templates.

Implement a template function that does filtering from tasks 2.2 2.3 to a container. Test it with vectors and sets.

`std::remove_if` doesn't work with sets, so to make your template function universal only use **`std::copy_if`**.

Task 5 (optional). Accumulate.

Use **`std::accumulate`** to compute the total Fare of all passengers.

This function is intended to be used with values that could be summed, so you need to create a new **`Passenger`** object to store the sum and overload corresponding operators/functions treating **`Passenger`** objects as mathematical objects.

Task 6. Custom dynamic container.

Create your own container class, **`PassengerRegister`**.

1. It should store passengers in a map **`std::map<int, Passenger*>`** . The key of the map is the id of the passenger. The value of the map is a **pointer** to a passenger object, not the object itself.

2. Create a method **`addPassenger(const Passenger& p)`** in the class. It should create a new passenger by using operator **`new`** and add the pointer to the map.

```
Passenger* passengerPointer = new Passenger(p);
```

3. Create a method **`freeAll()`** in the class. It should remove all passengers from the map and free the memory using operator **`delete`**.

```
delete passengerPointer;
```

4. Create a method **`getPassenger(int id)`** that returns a pointer to the passenger with the given id if it exists in the map. Return **`nullptr`** if the employee doesn't exist.

5. Implement a destructor for the class **`PassengerRegister`** and free all memory there.

Test your program by filling your container with passengers from the vector and retrieving passenger pointers from it.