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

Introduction to Programming, workshops 25-26.

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

GitHub: https://github.com/dsba-z/week13cpp2021

Replit: https://replit.com/@l8doku/Workshop23FunctionObjects

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.

Task 1. Applying functions to collections.

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

Task 2. Filtering.

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 (additional).

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 (additional). 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 5. Templates.

Implement a template function that does filtering from tasks 4.2 4.3 to a container. Test it with vectors and sets.