Задача

Решение задачи можно присылать (предпочтительнее ссылка на github, и т. п.) на vkonovodov@gmail.com. Бонусные баллы за решения задач могут быть поставлены только нескольким первым приславшим правильное решение.

Решение должно быть оформлено в виде компилирующегося кода (можно использовать C++11/14/17/20)

В проекте на C++ определены такие типы:

```
#include <ctime>
#include <vector>
struct TItem {
    int value;
    time_t timestamp;

    TItem(int v)
        : value(v)
        , timestamp(std::time(0)) {}
    // ...
};
using Items = std::vector<TItem>;
```

Для целей тестирования кода разработчику потребовалось написать функции MakeItemsSimple, создающие примеры объектов Items в небольших тестах следующим образом:

```
    Items items = MakeItemsSimple<0, 1, 4, 5, 6>();
    Items newItems = MakeItemsSimple<7, 15, 1>();
```

Эти функции создают векторы Items и заполняют их значениями value, переданными в параметрах шаблона (вызовом определенного выше конструктора).

Кроме того, для тестирования разработчику требуется функция MakePredicate, создающая лямбда-объект, который проверяет, что в векторе есть TItem с переданным значением value, и имеющая такой интерфейс использования:

```
#include <assert.h>
// ....
    auto isFound = MakePredicate(items);
    auto isFoundNew = MakePredicate(newItems);
    assert(isFound(0) == true);
    assert(isFound(7) == false);
    assert(isFoundNew(7) == true);
    assert(isFoundNew(6) == false);
```

Помогите разработчику — реализуйте при помощи шаблонов с переменным количеством аргументов функцию MakeItemsSimple, а также реализуйте *без использования* циклов функцию MakePredicate.