Creative Software Programming Practice (week-12-1)

Every assignment will be announced on **Thursday** and should be sumitted by next **Tuesday**.

We have practice classes on Wednesdays and Thursdays.
The contents of the practice class are different from the assignments and aim to be completed on the same day.

Assignments will be publish on Thursday and must be submitted by the next Tuesday.

In this week **Handed out will be Nov 19, 2020, Due Nov 26, 2020**

Topics

1. Practices-1

1. Practices-1

Implement the program to pass the test case written in `p1.cc` below.

You can declare additional functions, but you cannot declare variables.

- `Array + T` push_back T to Array.
- `Array += T` push_front T to Array.
- `Array - T` finds T in Array and removes it. Look from the front.
- `Array -= T` finds T in Array and removes it. Look from behind.
- `Array * T` multiplies all elements in Array by T.
- `Array / T` divides all elements of Array by T.
- `Array[` I `]` returns the `I-th` element of Array. Invalid I is not given.

- week-12
    - p1.cc
    - p1.h

```
// p1.h
#include <deque>
#include <functional>
#include <algorithm>
template <typename T>
class Array {
    std::deque<T> _arr;

public:
    Array() {}

    T& operator[](size_t index);

    size_t size() const {
        return _arr.size();
    }

    void view(const std::function<void(const T&)>& f) {
```

```
        for (auto t : _arr) {
            f(t);
        }
    }
};

// p1.cc
#include <iostream>
#include <cassert>
#include "p1.h"

int main() {
    auto array = Array<float>();

    array + 3.f;
    assert(array.size() == 1);
    assert(array[0] == 3.f);

    array += 1.f;
    assert(array.size() == 2);
    assert(array[0] == 1.f);

    array + 5.f + 3.f + 7.f + 3.f;
    assert(array.size() == 6);
    assert(array[2] == 5.f &&
           array[3] == 3.f &&
           array[4] == 7.f &&
           array[5] == 3.f);

    array - 3.f;
    assert(array.size() == 5);
    assert(array[1] == 5.f);

    array -= 3.f;
    assert(array.size() == 4);
    assert(array[3] == 7.f);

    assert(array[0] == 1.f &&
           array[1] == 5.f &&
           array[2] == 3.f &&
           array[3] == 7.f);

    array * 2.f;
    assert(array[0] == 2.f &&
           array[1] == 10.f &&
           array[2] == 6.f &&
           array[3] == 14.f);

    array / 2.f;
    assert(array[0] == 1.f &&
           array[1] == 5.f &&
           array[2] == 3.f &&
           array[3] == 7.f);

    return 0;
}
```