Create the following custom arrays:

* Immutable array - when adding/removing element a new object is created, the original is not changed

```
// Usage example
void testImmutableArray() {
    ImmutableArray<int> arr1;
    auto arr2 = arr1.add(10).add(20).add(30);
    auto arr3 = arr2.removeAt(1); // Remove element at index 1

    // arr1 is still empty, arr2 has [10,20,30], arr3 has [10,30]
    std::cout << "arr1 size: " << arr1.size() << std::endl; // 0
    std::cout << "arr2 size: " << arr2.size() << std::endl; // 3
    std::cout << "arr3 size: " << arr3.size() << std::endl; // 2
}
```

* Avg array - can provide the average sum of the elements of the array

```
// Usage example
void testAvgArray() {
    AvgArray<int> arr;
    arr.push_back(10);
    arr.push_back(20);
    arr.push_back(30);

    std::cout << "Average: " << arr.average() << std::endl; // 20
    std::cout << "Sum: " << arr.getSum() << std::endl; // 60

    arr.set(1, 25); // Change 20 to 25
    std::cout << "New average: " << arr.average() << std::endl; // 21.67
}
```

* Sum array - can provide the sum of the elements of the array

```
// Usage example
void testSumArray() {
    SumArray<int> arr;
    arr.push_back(5);
    arr.push_back(15);
    arr.push_back(25);

    std::cout << "Sum: " << arr.sum() << std::endl; // 45

    arr.insert(1, 10);
    std::cout << "Sum after insert: " << arr.sum() << std::endl; // 55
}
```

* Limited to n array - after n elements, the first one is pushed out when inserting a new one

```
// Usage example
void testLimitedArray() {
    LimitedArray<int> arr(3); // Maximum 3 elements

    arr.push_back(1);
    arr.push_back(2);
    arr.push_back(3);
    arr.push_back(4); // This pushes out 1

    std::cout << "Size: " << arr.size() << std::endl; // 3
    for (size_t i = 0; i < arr.size(); ++i) {
        std::cout << arr[i] << " "; // 2 3 4
    }
    std::cout << std::endl;
}
```

* History array - an array that can retrieve the last 10 changes

```
// Usage example
void testHistoryArray() {
    HistoryArray<int> arr;

    arr.push_back(10);
    arr.push_back(20);
    arr.push_back(30);

    std::cout << "Size: " << arr.size() << std::endl; // 3

    arr.set(1, 25); // Change 20 to 25
    std::cout << "After set: " << arr[1] << std::endl; // 25

    std::cout << "History size: " << arr.getHistorySize() << std::endl;
}
```

* Mutating array - when setting a value to an index, the value is insted summed, not replaced

```
// Usage example
void testMutatingArray() {
    MutatingArray<int> arr;

    arr.push_back(10);
    arr.push_back(20);
    arr.push_back(30);

    std::cout << "Initial: " << arr[1] << std::endl; // 20

    arr.set(1, 5); // Add 5 to existing value
    std::cout << "After set(1, 5): " << arr[1] << std::endl; // 25
```

```
    arr.set(1, 10); // Add 10 to existing value
    std::cout << "After set(1, 10): " << arr[1] << std::endl; // 35

    arr.set(5, 100); // Extends array and sets index 5 to 100
    std::cout << "Size after extending: " << arr.size() << std::endl; //
6
    std::cout << "Value at index 5: " << arr[5] << std::endl; // 100
}
```