Advanced C++ Programming

Assignment Sheet 1

Preliminaries

In the following, you will be working with 4 files. The `CMakeFile` states how the assignment is compiled. The main function and the tests are in the `VectorTest.cpp`. The `Vector` class, which you will have to complete has a header file `Vector.h` and a cpp file `Vector.cpp`. The completion of the `Vector` class is done only in the `Vector.cpp`. This is where your solution will be, which you upload once you are finished. Your solution has to work with the provided (and unchanged) `CMakeFile`, `VectorTest.cpp` and `Vector.h` files. If you are unable to compile, run and valgrind your code, you can use our online validator. When you are done, upload your solution (named as `Vector.cpp`!) to StudOn.

1 Vector Class

Implement a vector class in `Vector.cpp`. Your class has to pass the provided tests and it must not have any memory leaks.

- The project should compile as provided and execution will result into an error.
- Have a look at the `Vector.h` file. You will find the functions, which need to be implemented. Notice that the `operator[]` function and the private class members are given.
- Your Vector class shall contain the function `size()`, which returns the size of the Vector, `begin()` and `end()`, which can be used to iterate over the `data_` pointer and a `swap` function, which uses `std::swap` to swap two vectors.
- Implement all constructors, the destructor and satisfy the "rule of 5".
- Implement the arithmetic operators: +, -, *, +=, -=, *=, which perform said operation element-wise. Use `assert` to test that both vectors have matching sizes.
- Implement the (in)equality operators == and != .
- Vector instances should be printable to `std::ostream`. In the output, the values should be separated by one ore more spaces/tabs.
- vector instances should be readable from `std::istream`. The reader should at least support the format specified above. Check for errors and report with `std::runtime_error`.
- Implement the dot product using `std::accumulate`.
- Compile and run your project with the provided `CMakeFile`.

Make sure your implementation passes the provided tests! Run your program also with `-fsanitize=address` and with `valgrind`[1] to detect memory leaks or invalid accesses.

[1] https://valgrind.org/

Hint: In some environments `valgrind` may report false positives. You may want to check this by comparing the `valgrind` output of your program to the `valgrind` output of a minimal program containing only an empty main function.