1-Specification Comments(RMEs)

The interface of a function includes its *signature*, which consists of the function’ s name and parameter types. The return type of the function is also part of its interface.

Another part of a function’ s interface is documentation about what it does, as in the following:

```
1  //REQUIRES: v is not empty
2  //EFFECTS: returns the arithmetic mean of the numbers in vdouble
3  mean(std::vector<double> v);
```

The documentation format we use is an *RME*, which consists of a *REQUIRES* clause, a *MODIFIES* clause, and an *EFFECTS* clause.

1. REQUIRES Clause

The *REQUIRES* clause lists what the function requires in order to accomplish its task. If the requirements are not met, then the function provides no guarantees – the behavior is *undefined*, and anything the function does (e.g. crashing your computer, stealing your credit-card info, etc.) is valid. Thus, a user should never call a function with arguments or in a state that violates the REQUIRES clause.

Within the function definition, the implementation is allowed to assume that the REQUIRES clause is met – again, a user should never call the function if they are violated. A good practice is to *assert* that the REQUIRES clause is met.

If a function doesn’ t have any requirements, the REQUIRES clause may be elided. Such a function is called *complete*, while one that has requirements is called *partial*.

2. MODIFIES Clause

The *MODIFIES* clause specifies the entities outside the function that might be modified by it. This includes pass-by-reference parameters, global variables, and input and

output streams and so on.

The MODIFIES clause only specifies what entities may be modified, leaving out any details about what those modifications actually might be. The latter is the job of the EFFECTS clause. Instead, the purpose of the MODIFIES clause is for the user to quickly tell what items might be modified.

If no non-local entities are modified, the MODIFIES clause may be elided.

3. EFFECT Clause

🏅 The *EFFECTS* clause specifies what the function actually does. This includes details about what modifications are made, if any, as well as what the return value means, if there is one. All functions should have an EFFECTS clause – if a function does nothing, there is no point to writing the function.

The EFFECTS clause should generally only indicate *what* the function does without getting into implementation details (the *how*). It is part of the interface of a function, so it should not be affected if the implementation were to change.