Concurrent Data Structures Lab Assignment 1

Bernat Xandri Zaragoza

1. **Task 1:** Standard non-concurrent set.

Once the behaviour of the set data type has been understood, we can add a set of instructions in the form of events that will be allowed. You can find the events in the code attached in this hand-in.
We can check that the events are allowed by using the function test_events, and seeing that we don't get any invalid statements.

To run the code, I'm using the command:
g++ -std=c++17 -Wall -g -lpthread main.cpp -o Lab1

```
## Test 1
- add(1) -> 1
- add(1) -> 0
- rmv(1) -> 1

## Test 2
- add(1) -> 1
- add(2) -> 0 is invalid, since 1 was returned.
  - Test State: {1, 2}

## Test 3
- add(1) -> 1
- ctn(1) -> 1
- rmv(2) -> 1 is invalid, since 0 was returned.
  - Test State: {1}


# Valid sequence of 10+ instructions added
## Test 4
- add(1) -> 1
- add(2) -> 1
- add(3) -> 1
- add(4) -> 1
- add(5) -> 1
- add(1) -> 0
- add(5) -> 0
- add(3) -> 0
- rmv(1) -> 1
- rmv(2) -> 1
- rmv(3) -> 1
- rmv(4) -> 1
```

2. **Task 2**: A Simple Set.

Once all the methods have been implemented, we can observe the expected result, as described in the exercise. The 1 thread version runs correctly all 200 events, but the concurrent version with 4 threads fails. This is due to not having implemented any synchronization mechanism for the set.
I was expecting the fail to happen earlier, and it surprises me that it can reach 162 events without failing.

Here a capture of the execution results:

```
# Task 2: Simple Set
## Testing `SimpleSet` with 1 thread
Successfully validated 200 events

## Testing `SimpleSet` with 4 threads
- rmv(125) -> 0 is invalid, since 1 was returned.
  - Test State: {0, 1, 3, 4, 7, 11, 12, 13, 15, 16, 18, 20, 25, 28, 30, 31, 35, 42, 50, 51, 55, 56, 57, 68, 69, 70, 74, 76, 79, 84, 87
, 89, 92, 97, 107, 110, 112, 114, 116, 117, 118, 120, 126}
  - Concurrent State: SimpleSet {...}
Validation failed after 162 events
```

3. **Task 3**: A coarse-Grained Set.

The implementation of the coarse-grained set is pretty straight forward, since we only need to add the lock in every method. Since it's a shared mutex, we only need to make sure to lock and unlock in the correct place.
The result that we get is the following:

```
# Task 3: Coarse Set
## Testing `CoarseSet` with 4 thread and seed: 0
Successfully validated 200 events

## Testing `CoarseSet` with 4 thread and seed: 1
Successfully validated 200 events

## Testing `CoarseSet` with 4 thread and seed: 2
Successfully validated 200 events

## Testing `CoarseSet` with 4 thread and seed: 3
Successfully validated 200 events

## Testing `CoarseSet` with 4 thread and seed: 4
Successfully validated 200 events

## Testing `CoarseSet` with 4 thread and seed: 5
Successfully validated 200 events

## Testing `CoarseSet` with 4 thread and seed: 6
Successfully validated 200 events

## Testing `CoarseSet` with 4 thread and seed: 7
Successfully validated 200 events
```

4. **Task 4**: A fine-Grained Set.

The implementation for the fine-grained set was more difficult, since there were many corner cases to take into consideration. While the ctn method didn't require too much consideration, for the rmv and add we had to consider the multiple possible states of the linked list, as well as the different relations between the nodes.
I think I managed to write some code that works correctly, as it manages to pass the test units presented in the screen capture, but it is definitely not the best version of the code, since I ended up having to check so many specific scenarios one by one, that the code is very long (and probably not super-efficient). I'm sure my code could be generalized into a more universal implementation.

```
# Task 4: Fine Set
## Testing `FineSet` with 4 thread and seed: 0
Successfully validated 200 events

## Testing `FineSet` with 4 thread and seed: 1
Successfully validated 200 events

## Testing `FineSet` with 4 thread and seed: 2
Successfully validated 200 events

## Testing `FineSet` with 4 thread and seed: 3
Successfully validated 200 events

## Testing `FineSet` with 4 thread and seed: 4
Successfully validated 200 events

## Testing `FineSet` with 4 thread and seed: 5
Successfully validated 200 events

## Testing `FineSet` with 4 thread and seed: 6
Successfully validated 200 events

## Testing `FineSet` with 4 thread and seed: 7
Successfully validated 200 events
```