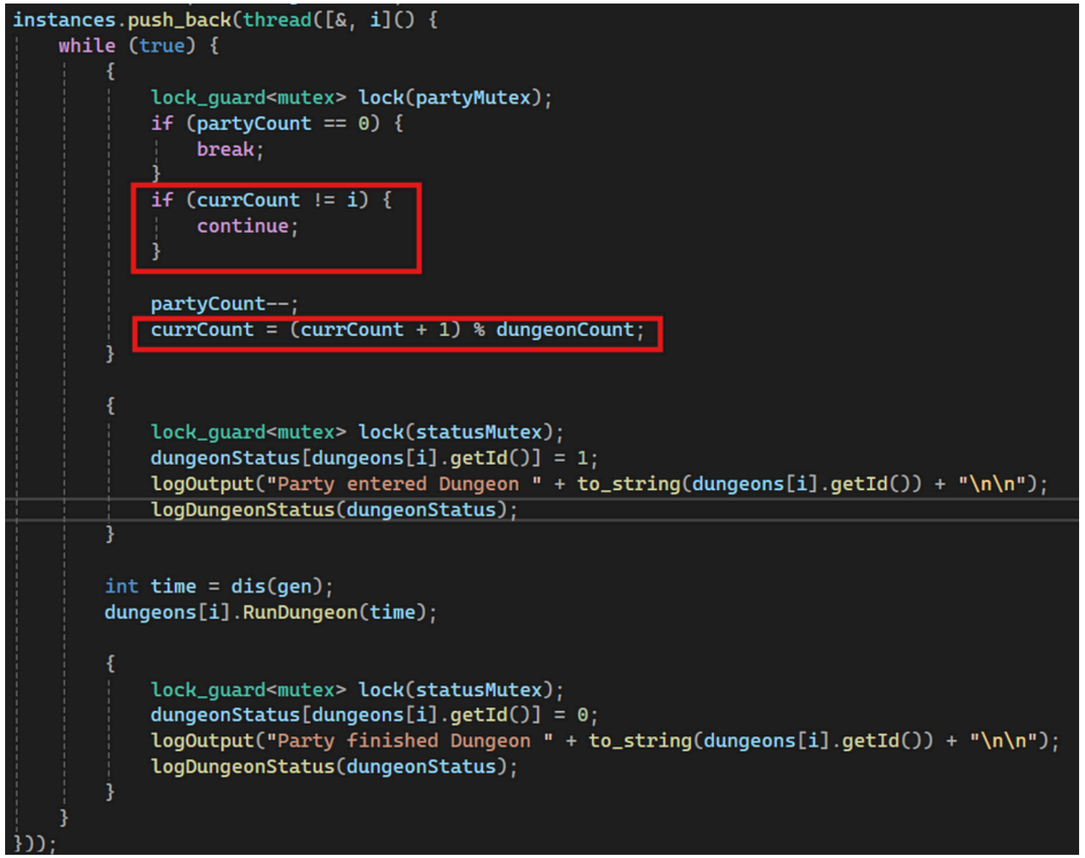
Possible Deadlock

- Mutexes were used to protect shared resources in the code
- Two mutexes were used:
  - partyMutex
  - statusMutex
- There shouldn’t be any deadlocks as they are independent from each other.

```
instances.push_back(thread([&, i]() {
    while (true) {
        {
            lock_guard<mutex> lock(partyMutex);
            if (partyCount == 0) {
                break;
            }
            if (currCount != i) {
                continue;
            }

            partyCount--;
            currCount = (currCount + 1) % dungeonCount;
        }

        {
            lock_guard<mutex> lock(statusMutex);
            dungeonStatus[dungeons[i].getId()] = 1;
            logOutput("Party entered Dungeon " + to_string(dungeons[i].getId()) + "\n\n");
            logDungeonStatus(dungeonStatus);
        }

        int time = dis(gen);
        dungeons[i].RunDungeon(time);

        {
            lock_guard<mutex> lock(statusMutex);
            dungeonStatus[dungeons[i].getId()] = 0;
            logOutput("Party finished Dungeon " + to_string(dungeons[i].getId()) + "\n\n");
            logDungeonStatus(dungeonStatus);
        }
    }
}));
```

Possible Starvation

- In allocating parties to dungeons, a round robin queueing system was implemented. The code checks if it's the current dungeon's turn and if the dungeon is empty. Only then the party will be assigned to the dungeon. Otherwise, the code will skip to the next dungeon.
- There might be a discrepancy in the number of parties served. But, the total time served should be about the same.



```
instances.push_back(thread([&, i]() {
    while (true) {
        {
            lock_guard<mutex> lock(partyMutex);
            if (partyCount == 0) {
                break;
            }
            if (currCount != i) {
                continue;
            }

            partyCount--;
            currCount = (currCount + 1) % dungeonCount;
        }

        {
            lock_guard<mutex> lock(statusMutex);
            dungeonStatus[dungeons[i].getId()] = 1;
            logOutput("Party entered Dungeon " + to_string(dungeons[i].getId()) + "\n\n");
            logDungeonStatus(dungeonStatus);
        }

        int time = dis(gen);
        dungeons[i].RunDungeon(time);

        {
            lock_guard<mutex> lock(statusMutex);
            dungeonStatus[dungeons[i].getId()] = 0;
            logOutput("Party finished Dungeon " + to_string(dungeons[i].getId()) + "\n\n");
            logDungeonStatus(dungeonStatus);
        }
    }
}));
```