```
agent<action> MakeAgent(int x, int y){
    agent<action> newAgent;
    newAgent.x = x;
    newAgent.y = y;
    newAgent.Actions[0] = new moveLeft;
    newAgent.Actions[1] = new moveRight;
    newAgent.Actions[2] = new moveUp;
    newAgent.Actions[3] = new moveDown;
    return newAgent;
}
state<action> MakeWorld(){
    state<action> newWorld;
    newWorld.CurrentAgent = 0;
    newWorld.ProtagonistID = 0;
    newWorld.Goal = MakeAgent(1000,0);
    newWorld.Agents[0] = MakeAgent(0,0);
    newWorld.Agents[1] = MakeAgent(-1000,0);
    newWorld.Obstacles[0] = MakeAgent(1,0);
    newWorld.Obstacles[1] = MakeAgent(0,1);
    return newWorld;
}

void BenchmarkSerialAlgorithm(int depth){
    double timeBefore = omp_get_wtime();
    expectimax(MakeWorld(), depth);
    double timeAfter = omp_get_wtime() - timeBefore;
    cout << "Serial Execution Time of Expectimax with depth " << depth << " is " <<
    timeAfter << "s." << endl;
}

void BenchmarkParallelAlgorithm(int depth, int numThreads){
    double timeBefore = omp_get_wtime();
    expectimaxParallel(MakeWorld(), depth, numThreads);
    double timeAfter = omp_get_wtime() - timeBefore;
    cout << "Parallel Execution Time of Expectimax with depth " << depth << " and " <<
    numThreads << "threads is " << timeAfter << "s." << endl;
}
```