Task 1:

Q1:

a0: 8 9 10 11 11 12
a1: 9 9 10 11

Q2:

a0: 8 9 10 11 12 12 12 12 12 12 11 12
a1: 9 10 11

Explanation: I added a private helper function "int getLargestConstraintTimestep(int agent_id, const list<Constraint>& constraints) const;" that returns the largest timestep present in the constraint for a specific agent and when the current timestep is smaller or equal to the largest constraint timestep, I do not count it as reaching the goal.

Q3:

Constraints:

```
constraints.push_back(make_tuple(1, 10, 11, 2));
constraints.push_back(make_tuple(1, 10, -1, 2));
constraints.push_back(make_tuple(1, 9, -1, 2));
```

a0: 8 9 10 11 12
a1: 9 10 17 10 11
Sum of cost: 8

Task 2:

Q1:

a0: 8 9 10 11 12
a1: 9 10 17 10 11 11
Sum of cost: 9

Explanation: I iterated over the paths and I made each location in each path a constraint with the appropriate timestep (which is the index).

Q2:

a0: 9 10 11
a1: 8 9 10 17 18 19 12
Sum of cost: 8

Explanation: I set a flag in task2.cpp where timestemp == -1 indicates the goal state and it is always checked in AStarPlanner.cpp regardless of the other agent's timestep.

Q3:

It didn't behave as expected (Agent 0 went over Agent 1 when Agent 1 reached its goal state) mainly because I hard-coded the agent IDs. It also went on forever instead of stopping when a path couldn't be found, so I manually set a threshold to prevent the code from looping forever.

Task 3:

Q1:

Sum of cost:

8,8,8,9,17,8,13,14,11,12,73,74,64,79,71,74,77,61,50,69,75,63,77,62,84,
59,77,49,62,71,53,69,81,60,70,60,77,76,70,74,69,83,66,68,74,67,61,68,7

```
1,59
        CPU time:

0,0,0,1,12,1,59,105,3,137,7,39,8,97,4,256,66,21,2,1,26,62,28,1,7,18,19
0,3,5,52,2,2,59,8,11,2,3,21,16,1,46,21,3,1,8,3,17,297,18,17
    Q2:
        Sum of cost (* == no solution):

8,*,*,*,17,*,*,*,11,12,*,*,66,87,*,78,80,68,55,69,77,71,78,74,92,63,79
,52,70,78,57,71,96,63,75,73,81,79,70,78,74,84,68,74,80,74,67,69,83,70
        CPU time:

0,0,0,0,0,1,1,0,0,0,44,6,21,46,67,34,50,28,14,22,26,31,28,32,52,22,31,
12,24,42,13,34,39,17,25,34,31,32,20,27,27,28,20,36,21,27,51,30,39,36
```