**Dieudonne Muhirwa M14185704**

**Allen Detmer M15216736**

**JiaBin Chang M13782945**

**Joseph Matawaran M15228497**

**Contributions**

**We did this assignment through mob programming three times a week. And finished the word file all together. The contribution of all groupmates is even.**

**20CS6033**

**Instructor: A. Ralescu**

**Fall 2022**

**Assignment 5**

**Assigned 10/24/2022**

**Due on Canvas 11/08/2022, at 11:59PM**

**50 points**

Implementation of blocks’ world problem. Below are the various test cases and their respective results. The paths from a start to a goal node are maintained as lists.

**TEST CASE 1:** Valid start state with goal state in a different sequence in the array

|  |  |  |  |
| --- | --- | --- | --- |
| **START STATE** | |  | **GOAL STATE** |
|  |  |  |  |
|  |  |  |  |
| b |  |  | b |
| a |  |  | a |

***Start state:***

start([[on, b, a], [clear, b], [on, a, "table"]]).

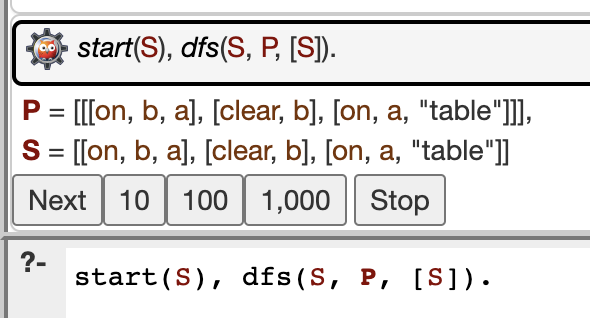
***Goal state:***

goal([[on, b, a], [on, a, "table"], [clear, b]]).

***Result:***

**start(S)** loads the start state to variable S

**dfs(S, P, [S])** returns the Path (P) from the start (S) to the goal states



***Conclusion:* Test case 1 was executed to test the goal state permutation and there is no recursion because the start state is different permutation to the goal state. Therefore, the arrangement of the elements in the list can be different from the order in the expected goal because it checks permutations.**

**TEST CASE 2:** 2-block problem

|  |  |  |  |
| --- | --- | --- | --- |
| **START STATE** | |  | **GOAL STATE** |
|  |  |  |  |
|  |  |  |  |
|  |  |  | b |
| a | b |  | a |

***Start state:***

start([[on, a, "table"], [on, b, "table"], [clear, a], [clear, b]]).

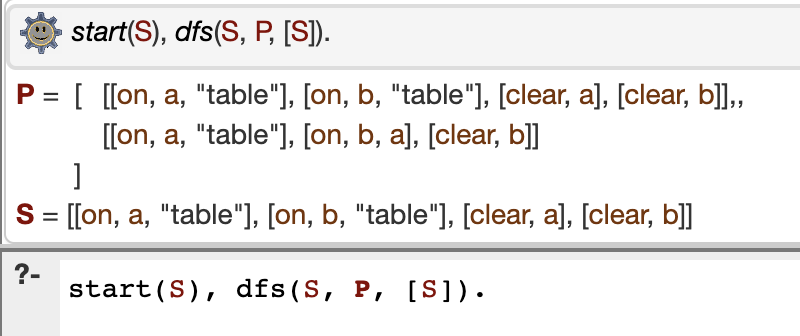
***Goal state:***

goal([[on, b, a], [on, a, "table"], [clear, b]]).

***Result:***

**start(S)** loads the start state to variable S

**dfs(S, P, [S])** returns the Path (P) from the start (S) to the goal states



***Conclusion:* Test case 2 was executed shorter than test cases 3 and 4 because it has less blocks involved and less branches in path (P). In this test case, it was the simplest case for DFS because the block was moved only once to achieve the goal state. Also, the arrangement of the elements in the list can be different from the order in the expected goal because it checks permutations.**

**TEST CASE 3:** 3-block problem

|  |  |  |  |
| --- | --- | --- | --- |
| **START STATE** | |  | **GOAL STATE** |
|  |  |  |  |
|  |  |  | c |
| c |  |  | b |
| a | b |  | a |

***Start state:***

start([[on, c, a], [on, a, "table"], [on, b, "table"], [clear, c], [clear, b]]).

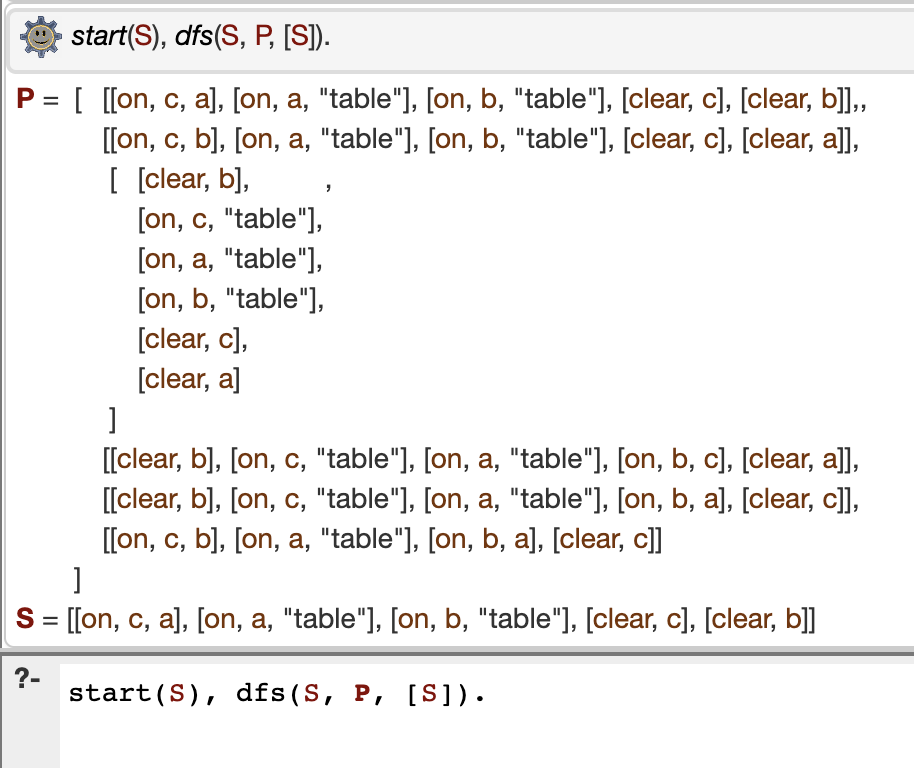
***Goal state:***

goal([[on, c, b], [on, b, a], [on, a, "table"], [clear, c]]).

***Result:***

**start(S)** loads the start state to variable S

**dfs(S, P, [S])** returns the Path (P) from the start (S) to the goal states



***Conclusion:* Test case 3 was executed longer than test cases 1 and 2 because it has more blocks and branches inside the path (P). In this case, there are several branches involved and it cannot simply enumerate all the branches. By solving this test case, it proves that recursive DFS algorithm is working. Also, the arrangement of the elements in the list can be different from the order in the expected goal because it checks permutations.**

**TEST CASE 4: 4-block problem**

|  |  |  |  |
| --- | --- | --- | --- |
| **START STATE** | |  | **GOAL STATE** |
|  |  |  | d |
|  |  |  | a |
| a | c |  | c |
| b | d |  | b |

***Start state:***

start([[on, a, b],[on, b, "table"], [on, c, d], [clear, c], [clear, a], [on, d, "table"]]).

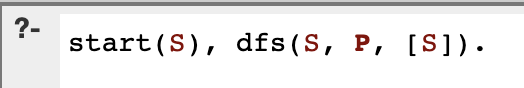
***Goal state:***

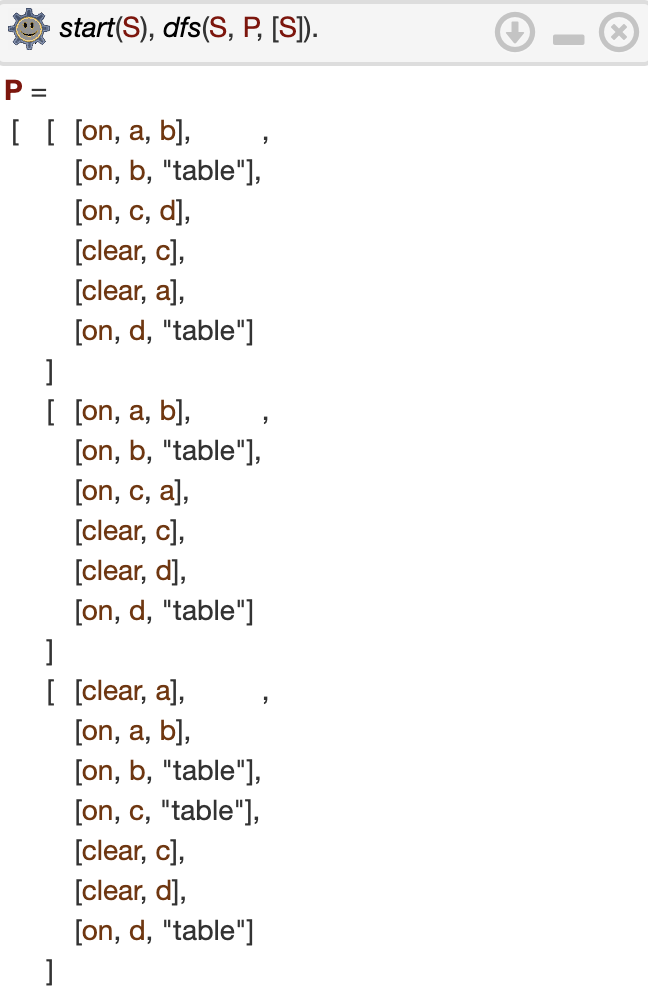
goal([[on, d, a], [on, a, c], [on, c, b], [on, b, "table"], [clear, d]]).

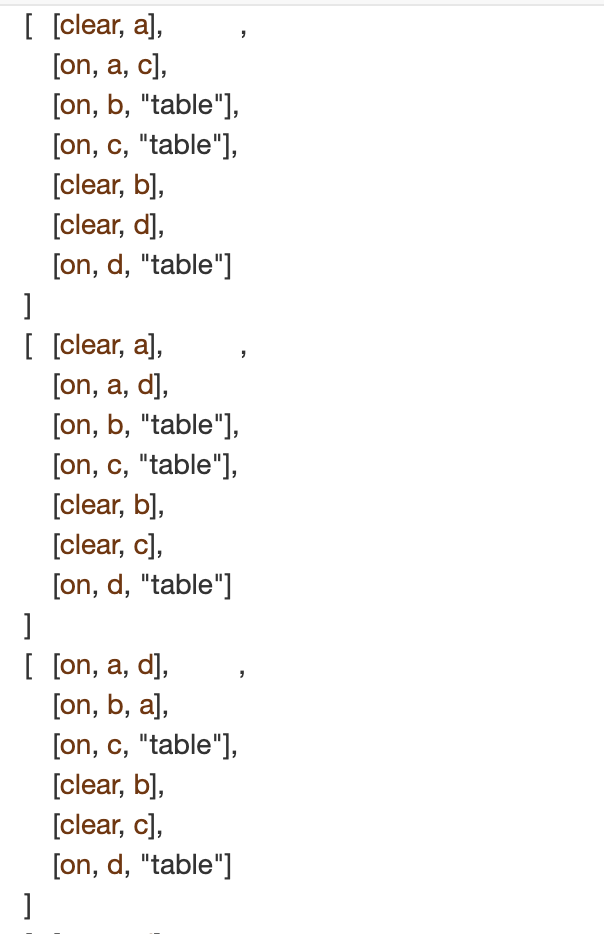
***Result:***

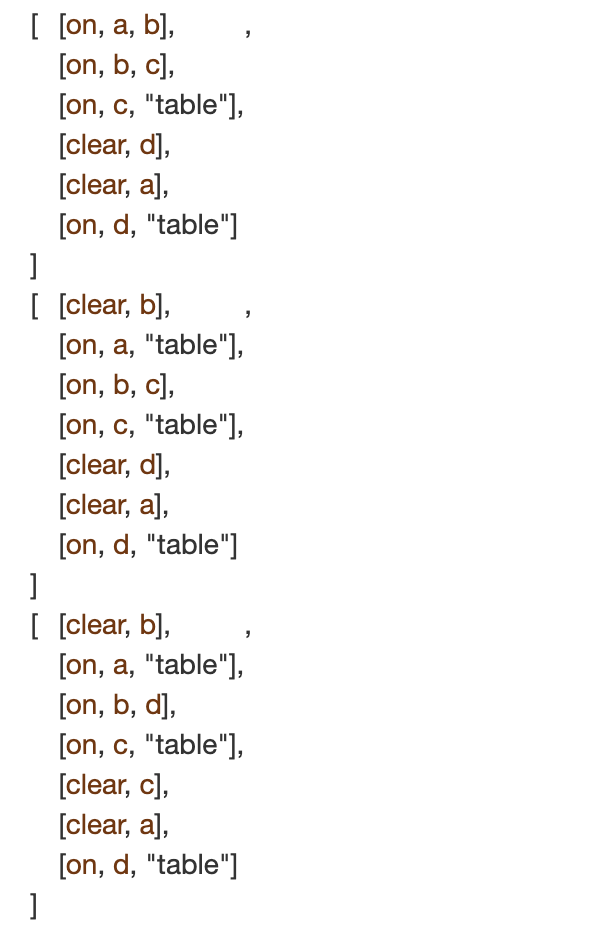
**start(S)** loads the start state to variable S

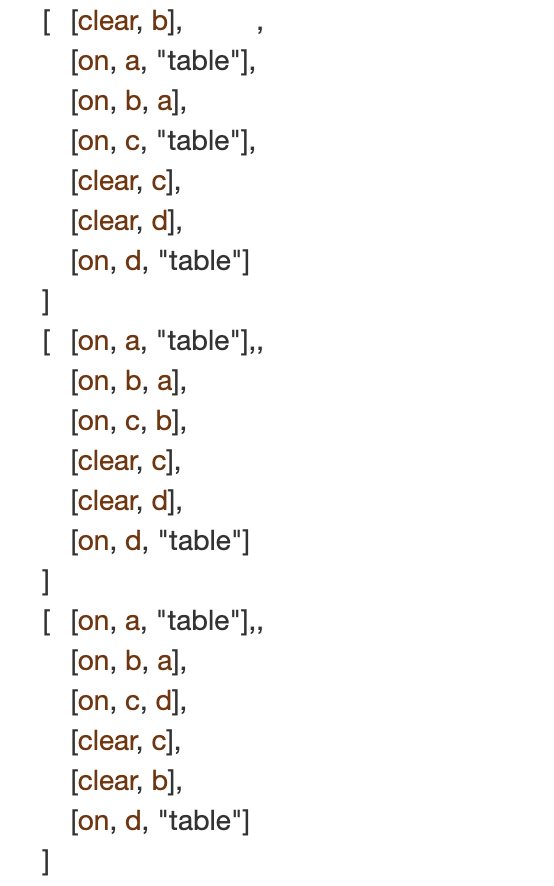
**dfs(S, P, [S])** returns the Path (P) from the start (S) to the goal states

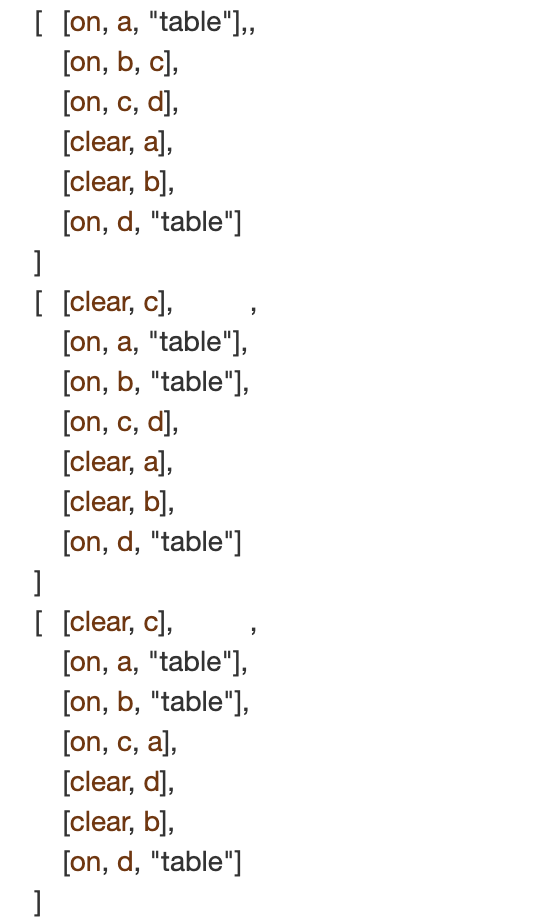


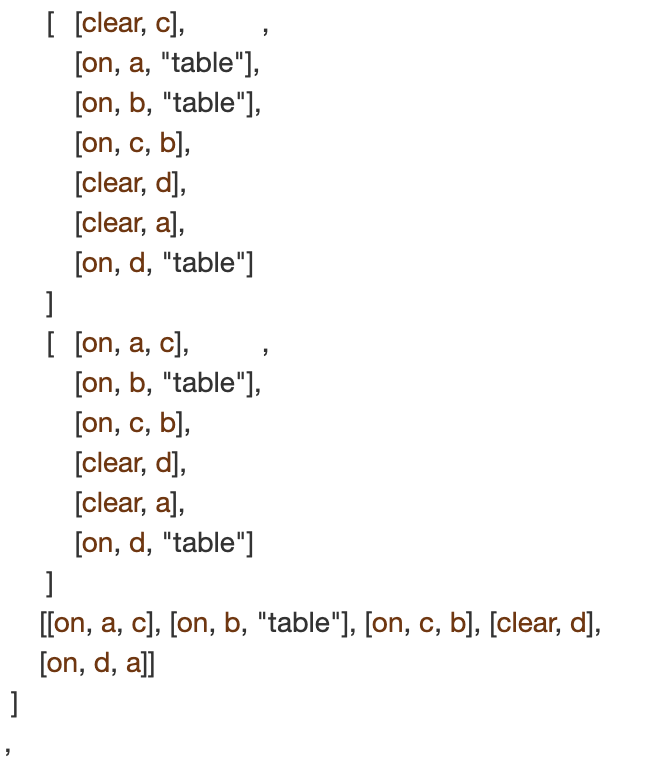


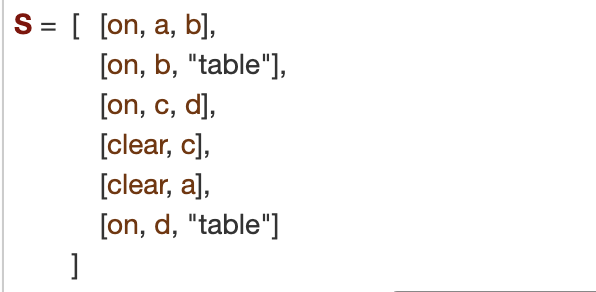












***Conclusion:* Test case 4 was the longest to be executed because it has the largest number of blocks and has 4 combinations. Moreover, the 4-block problem has more branches to visit. Also, the arrangement of the elements in the list can be different from the order in the expected goal because it checks permutations.**