**AI Lab – 4**

Name: Mokshit Oswal

Class: AI-B

Roll Number – 32

PRN: 12111304

Batch B2

**Problem Statement: Implement Goal Stack Planning in C/C++/Java.**

CODE (IN JAVA):

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

import java.util.Stack;

class Block {

char name;

boolean onTable;

boolean clear;

boolean held;

public Block(char name) {

this.name = name;

this.onTable = true;

this.clear = true;

this.held = false;

}

}

class BlockWorld {

private List<Block> blocks;

private Stack<String> plan;

public BlockWorld(int n, String initialState, String goalState) {

initializeBlocks(n);

populateInitialState(initialState);

this.plan = new Stack<>();

generatePlan(goalState);

}

private void initializeBlocks(int n) {

blocks = new ArrayList<>();

for (char c = 'A'; c < 'A' + n; c++) {

blocks.add(new Block(c));

}

}

private void populateInitialState(String initialState) {

String[] predicates = initialState.split("\\^");

for (String predicate : predicates) {

if (predicate.contains("on")) {

String[] tokens = predicate.replaceAll("[() ]", "").split("on");

char topBlock = tokens[0].charAt(0);

char bottomBlock = tokens[1].charAt(0);

blocks.stream().filter(block -> block.name == topBlock).findFirst().ifPresent(block -> {

block.onTable = false;

block.clear = false;

});

blocks.stream().filter(block -> block.name == bottomBlock).findFirst().ifPresent(block -> {

block.clear = false;

});

} else if (predicate.contains("ontable")) {

char block = predicate.replaceAll("[() ]", "").charAt(8);

blocks.stream().filter(b -> b.name == block).findFirst().ifPresent(b -> {

b.onTable = true;

b.clear = true;

});

} else if (predicate.contains("clear")) {

char block = predicate.replaceAll("[() ]", "").charAt(6);

blocks.stream().filter(b -> b.name == block).findFirst().ifPresent(b -> {

b.clear = true;

});

} else if (predicate.contains("hold")) {

char block = predicate.replaceAll("[() ]", "").charAt(5);

blocks.stream().filter(b -> b.name == block).findFirst().ifPresent(b -> {

b.held = true;

});

}

}

}

private void generatePlan(String goalState) {

Stack<String> subGoals = new Stack<>();

subGoals.push(goalState);

while (!subGoals.isEmpty()) {

String subGoal = subGoals.pop();

if (subGoal.contains("^")) {

String[] goals = subGoal.split("\\^");

for (String goal : goals) {

subGoals.push(goal);

}

} else {

boolean satisfied = false;

for (Block block : blocks) {

if (subGoal.contains("ontable") && block.onTable) {

plan.push("(putdown " + block.name + ")");

plan.push("(hold " + block.name + ")");

block.onTable = false;

block.clear = true;

block.held = false;

satisfied = true;

break;

} else if (subGoal.contains("clear") && block.clear) {

char blockName = subGoal.charAt(6);

if (block.name != blockName) {

int topBlockIndex = findBlockOnTop(blockName);

if (topBlockIndex != -1) {

Block topBlock = blocks.get(topBlockIndex);

plan.push("(unstack " + topBlock.name + " " + block.name + ")");

plan.push("(on " + topBlock.name + " " + block.name + ")^(clear " + topBlock.name + ")^(AE)");

topBlock.onTable = false;

topBlock.clear = false;

topBlock.held = true;

block.clear = false;

block.held = false;

satisfied = true;

break;

}

}

} else if (subGoal.contains("hold") && block.held) {

char blockName = subGoal.charAt(5);

if (block.name != blockName) {

int topBlockIndex = findBlockOnTop(blockName);

if (topBlockIndex != -1) {

Block topBlock = blocks.get(topBlockIndex);

plan.push("(unstack " + topBlock.name + " " + block.name + ")");

plan.push("(on " + topBlock.name + " " + block.name + ")^(clear " + topBlock.name + ")^(AE)");

topBlock.onTable = false;

topBlock.clear = false;

topBlock.held = true;

block.clear = false;

block.held = false;

satisfied = true;

break;

}

}

}

}

if (!satisfied) {

subGoals.push(goalState);

plan.clear();

for (Block block : blocks) {

if (block.held) {

plan.push("(putdown " + block.name + ")");

plan.push("(hold " + block.name + ")");

block.onTable = true;

block.clear = true;

block.held = false;

break;

}

}

}

}

}

}

private char findBlockOnTop(char blockName) {

for (int i = 0; i < blocks.size(); i++) {

Block block = blocks.get(i);

if (!block.onTable && block.clear) {

char topBlock = findBlockOnTop(block.name);

if (topBlock == blockName) {

return block.name; // Return the name of the top block

}

}

}

return ' '; // Return a sentinel value (e.g., space) to indicate no block on top

}

public void printPlan() {

System.out.println("Steps taken:");

while (!plan.isEmpty()) {

System.out.println(plan.pop());

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter the number of blocks: ");

int n = sc.nextInt();

sc.nextLine();

System.out.print("Enter the start state: ");

String initialState = sc.nextLine();

System.out.print("Enter the goal state: ");

String goalState = sc.nextLine();

BlockWorld blockWorld = new BlockWorld(n, initialState, goalState);

blockWorld.printPlan();

}

}

**OUTPUT SCREENSHOTS:**

A screenshot of a computer program

Description automatically generated



