1.

Input:

move the object that is 4 cm to the left, another 2 cm to the left

Output:

public void moveObjectLeft4Then2() {

this.returnToOrigin(); // Ensure the robot starts at the origin

this.moveX(-4); // Move 4 cm to the left

this.moveZ(0); // Assume the object is at ground level

this.openGripper(); // Open the gripper before lowering

this.moveY(-2); // Lower the arm 2 cm to reach the object

this.closeGripper(); // Grip the object

this.moveY(2); // Lift the object 2 cm

this.moveX(-2); // Move another 2 cm to the left

this.moveY(-2); // Lower the arm to ground level

this.openGripper(); // Release the object

this.moveY(2); // Lift the arm back up

this.returnToOrigin(); // Return to the origin

}

Test:

@Test

public void robotMovesObjectLeft4Then2FromNegativeX() {

Robot r = new Robot(-1, 2, 1);

r.moveObjectLeft4Then2();

ArrayList<Double> expectedPosition = new ArrayList<>(Arrays.asList(0.0, 2.0, 0.0));

assertEquals(expectedPosition, r.getPosition());

}

Result: Passed.

2.

Input:

“"pickup object at 3,0,9 and do not return to origin"

Output:

public void pickUpObject3\_0\_9Remain() {

this.returnToOrigin(); // Ensure the robot starts at the origin

this.moveX(3); // Move 3 cm to the right

this.moveZ(9); // Move 9 cm forward

this.openGripper(); // Open the gripper

this.moveY(-2); // Lower the arm 2 cm to reach the object

this.closeGripper(); // Close the gripper to pick up the object

this.moveY(2); // Lift the object 2 cm

// Do not return to origin

}

Test:

@Test

public void pickUpObject3\_0\_9Remain() {

Robot r = new Robot();

r.pickUpObject3\_0\_9Remain();

ArrayList<Double> expectedPosition = new ArrayList<>(Arrays.asList(3.0, 2.0, 9.0));

assertEquals(expectedPosition, r.getPosition());

}

Result: Passed.

3.

Input:

"pickup object at 4,0,3"

Output:

public void pickUpObject4\_0\_3() {

this.returnToOrigin(); // Ensure the robot starts at the origin

this.moveX(4); // Move 4 cm to the right

this.moveZ(3); // Move 3 cm forward

this.openGripper(); // Open the gripper

this.moveY(-2); // Lower the arm 2 cm to reach the object

this.closeGripper(); // Close the gripper to pick up the object

this.moveY(2); // Lift the object 2 cm

this.returnToOrigin(); // Return to the origin

}

Test:

@Test

public void robotPicksUpObjectAt4\_0\_3FromNonOrigin() {

Robot r = new Robot(1, 2, 1);

r.pickUpObject4\_0\_3();

ArrayList<Double> expectedPosition = new ArrayList<>(Arrays.asList(0.0, 2.0, 0.0));

assertEquals(expectedPosition, r.getPosition());

}

Result: Passed.

4.

Input:

“pickup object that is 7 cm to the left and 4 cm forward of origin”

Output:

public void pickUpObjectLeft7Forward4() {

this.returnToOrigin(); // Ensure the robot starts at the origin

this.moveX(-7); // Move 7 cm to the left

this.moveZ(4); // Move 4 cm forward

this.openGripper(); // Open the gripper before lowering

this.moveY(-2); // Lower the arm 2 cm to reach the object

this.closeGripper(); // Grip the object

this.moveY(2); // Lift the object 2 cm

this.returnToOrigin(); // Return to the origin

}

Test:

@Test

public void robotPicksUpObjectLeft7Forward4FromOrigin() {

Robot r = new Robot();

r.pickUpObjectLeft7Forward4();

ArrayList<Double> expectedPosition = new ArrayList<>(Arrays.asList(0.0, 2.0, 0.0));

assertEquals(expectedPosition, r.getPosition());

}

Result: Passed.

5.

Input:

“move the object from 5,0,1 to 3,2,1. do not return to origin”

Output:

public void moveObject5\_0\_1To3\_2\_1() {

this.returnToOrigin(); // Ensure the robot starts at the origin

this.moveX(5); // Move to the object's starting x position

this.moveZ(1); // Move to the object's starting z position

this.openGripper(); // Open the gripper before lowering

this.moveY(-2); // Lower the arm to ground level

this.closeGripper(); // Grip the object

this.moveY(2); // Lift the object

this.moveX(-2); // Move to the target x position

this.moveY(2); // Move to the target y position

this.moveY(-2); // Lower the arm to ground level

this.openGripper(); // Release the object

this.moveY(2); // Lift the arm

// Do not return to origin

}

Test:

@Test

public void robotMovesObjectFrom5\_0\_1To3\_2\_1FromNonOrigin() {

Robot r = new Robot(4, 4, 4);

r.moveObject5\_0\_1To3\_2\_1();

ArrayList<Double> expectedPosition = new ArrayList<>(Arrays.asList(3.0, 4.0, 1.0));

assertEquals(expectedPosition, r.getPosition());

}

Result: Passed.

6.

Input:

“pickup the object situated 2 cm to the right”

Output:

public void pickUpObjectRight2() {

this.returnToOrigin(); // Ensure the robot starts at the origin

this.moveX(2); // Move 2 cm to the right

this.openGripper(); // Open the gripper before lowering

this.moveY(-2); // Lower the arm 2 cm to reach the object

this.closeGripper(); // Grip the object

this.moveY(2); // Lift the object 2 cm

this.returnToOrigin(); // Return to the origin

}

Test:

@Test

public void robotPicksUpObjectRight2FromNonOrigin() {

Robot r = new Robot(1, 2, 1);

r.pickUpObjectRight2();

ArrayList<Double> expectedPosition = new ArrayList<>(Arrays.asList(0.0, 2.0, 0.0));

assertEquals(expectedPosition, r.getPosition());

}

Result: Passed.

7.

Input:

“pickup the object situated 2 cm to the right and place it at position 5,0,5. do not return to origin”

Output:

public void pickUpObjectRight2Place5\_0\_5Remain() {

this.returnToOrigin(); // Ensure the robot starts at the origin

this.moveX(2); // Move 2 cm to the right

this.openGripper(); // Open the gripper before lowering

this.moveY(-2); // Lower the arm 2 cm to reach the object

this.closeGripper(); // Grip the object

this.moveY(2); // Lift the object 2 cm

this.moveX(3); // Move to the target x position (5 - 2 = 3)

this.moveZ(5); // Move to the target z position

this.moveY(-2); // Lower the arm to ground level

this.openGripper(); // Release the object

this.moveY(2); // Lift the arm

// Do not return to origin

}

Test:

@Test

public void robotPicksUpObjectRight2PlacesAt5\_0\_5RemainsFromNonOrigin() {

Robot r = new Robot(4, 4, 4);

r.pickUpObjectRight2Place5\_0\_5Remain();

ArrayList<Double> expectedPosition = new ArrayList<>(Arrays.asList(5.0, 2.0, 5.0));

assertEquals(expectedPosition, r.getPosition());

}

Restul: Passed

8.

Input:

“take the object situated 4 cm to the left and move it 5 cm to the right. do not return to origin”

Output:

public void pickUpObjectLeft4MoveRight5Remain() {

this.returnToOrigin(); // Ensure the robot starts at the origin

this.moveX(-4); // Move 4 cm to the left

this.openGripper(); // Open the gripper before lowering

this.moveY(-2); // Lower the arm 2 cm to reach the object

this.closeGripper(); // Grip the object

this.moveY(2); // Lift the object 2 cm

this.moveX(5); // Move 5 cm to the right

this.moveY(-2); // Lower the arm to ground level

this.openGripper(); // Release the object

this.moveY(2); // Lift the arm

// Do not return to origin

}

Test:

@Test

public void robotPicksUpObjectLeft4MovesRight5RemainsAtOrigin() {

Robot r = new Robot();

r.pickUpObjectLeft4MoveRight5Remain();

ArrayList<Double> expectedPosition = new ArrayList<>(Arrays.asList(1.0, 2.0, 0.0));

assertEquals(expectedPosition, r.getPosition());

}

Result: Passed.

9.

Input:

“move the object situated 2 cm to the left to origin.”

Output:

public void moveObjectLeft2ToOrigin() {

this.returnToOrigin(); // Ensure the robot starts at the origin

this.moveX(-2); // Move 2 cm to the left

this.openGripper(); // Open the gripper before lowering

this.moveY(-2); // Lower the arm to ground level

this.closeGripper(); // Grip the object

this.moveY(2); // Lift the object

this.returnToOrigin(); // Move to the origin

this.moveY(-2); // Lower the arm to ground level

this.openGripper(); // Release the object

this.moveY(2); // Lift the arm

this.returnToOrigin(); // Return to the origin

}

Test:

@Test

public void robotMovesObjectLeft2ToOriginFromNegativeX() {

Robot r = new Robot(-4, 2, 2);

r.moveObjectLeft2ToOrigin();

ArrayList<Double> expectedPosition = new ArrayList<>(Arrays.asList(0.0, 2.0, 0.0));

assertEquals(expectedPosition, r.getPosition());

}

Result: Passed

10.

Input:

“move the object from 3,0,3 forward 2 cm”

Output:

public void moveObject3\_0\_3Forward2() {

this.returnToOrigin(); // Ensure the robot starts at the origin

this.moveX(3); // Move to the object's x position

this.moveZ(3); // Move to the object's z position

this.openGripper(); // Open the gripper before lowering

this.moveY(-2); // Lower the arm to ground level

this.closeGripper(); // Grip the object

this.moveY(2); // Lift the object

this.moveZ(2); // Move 2 cm forward

this.moveY(-2); // Lower the arm to ground level

this.openGripper(); // Release the object

this.moveY(2); // Lift the arm

this.returnToOrigin(); // Return to the origin

}

Test:

@Test

public void robotMovesObject3\_0\_3Forward2FromOrigin() {

Robot r = new Robot();

r.moveObject3\_0\_3Forward2();

ArrayList<Double> expectedPosition = new ArrayList<>(Arrays.asList(0.0, 2.0, 0.0));

assertEquals(expectedPosition, r.getPosition());

}

Result: Passed.

11.

Input:

“Pickup last object placed and move it 2 cm to the left. do not return to origin”

Output:

public void pickUpLastObjectMoveLeft2Remain() {

// Assuming the last object was placed at 3, 0, 5 (from previous instruction)

this.moveX(3); // Move to the object's x position

this.moveZ(5); // Move to the object's z position

this.openGripper(); // Open the gripper before lowering

this.moveY(-2); // Lower the arm to ground level

this.closeGripper(); // Grip the object

this.moveY(2); // Lift the object

this.moveX(-2); // Move 2 cm to the left

this.moveY(-2); // Lower the arm to ground level

this.openGripper(); // Release the object

this.moveY(2); // Lift the arm

// Do not return to origin

}

Test:

@Test

public void robotPicksUpLastObjectMovesLeft2RemainsAtOrigin() {

Robot r = new Robot();

r.pickUpLastObjectMoveLeft2Remain();

ArrayList<Double> expectedPosition = new ArrayList<>(Arrays.asList(1.0, 2.0, 5.0));

assertEquals(expectedPosition, r.getPosition());

}

Result: Passed

12.

Input:

Ideas

* Pickup last object placed and move it 2 cm to the left - done
* Pick one object at cords.. and place it on top of the object at cords..
* Take the object at 1,0,1 and place it at 2,0,1. Then take another object at 3,0,3 and place it one cm to the right.
* Take one object situated at cords.. and place it on top of the object situated at cords ..
* Move an object from coords.. at coords… The pick it up and move it 2 cm to the left
* Pickup the last object moved to coords ..
* Pickup the object 4 cm to the left and drop it from 1 cm distance to the coords 4,0,2
* From origin, draw a sqare in the air with a side length of 3.
* There are 3 objects at coords 3,0,1, 4,0,2 and -2,0,2. I want you to pick them and place them on top of eachother at coordinates -5,0,5
* There are 2 objects are coordinates 3,0,2 and -3,0,2. Can you swap them around.