Introductory Robot Programming

Fall – 2020

Instructor:	Zeid Kootbally	**RWA#1:**	Task Planner
Email:	zeidk@umd.edu	**Due date:**	10/01/2020

1 Instructions

For this assignment you will need to plan a sequence of operations to be done by a robot in order to execute a manufacturing task.

- The plan is generated offline, i.e., you generate a plan first and then task the robot to follow the instructions in the plan. The assignment only focuses on plan generation and not on plan execution.
- The manufacturing task for this assignment is kitting (or kit building). Kitting is the process by which individually separate but related items are grouped, packaged, and supplied together as one unit. Frequently, these items are assembled into a kit and delivered to the assembly line for workers to integrate onto each individual final product.

We will assume we have a dual-arm gantry robot for this assignment. Each arm consists of a suction gripper to grasp parts from the bins and place them in a kit tray (located on an automated guided vehicle (AGV)). The workcell for this assignment is represented in Figure 1.

The key features for task planning in this context are:

- Initial state: State of the world when the simulation starts.
- Goal state: State of the world that you need to reach.
- Robot actions: Actions the robot can perform to go from the initial state to the goal state.

2 Initial State

The initial state represents the state of the world when your program starts. The initial state informs on the status of the following components, which will be hard coded in your program.

- You will need to use a data structure to represent the initial state.

2.1 Number of Parts in Bins

In this assignment you will use parts of the same type but of different colors (see Figure 2) and we can have a maximum of 10 parts of each color in each bin depicted in Figure 1.

2.2 Number of Parts in Kit Tray

In the initial state (when you start your program) the kit tray has:

- 0 **red** part.
- 0 **green** part.
- 0 **blue** part.

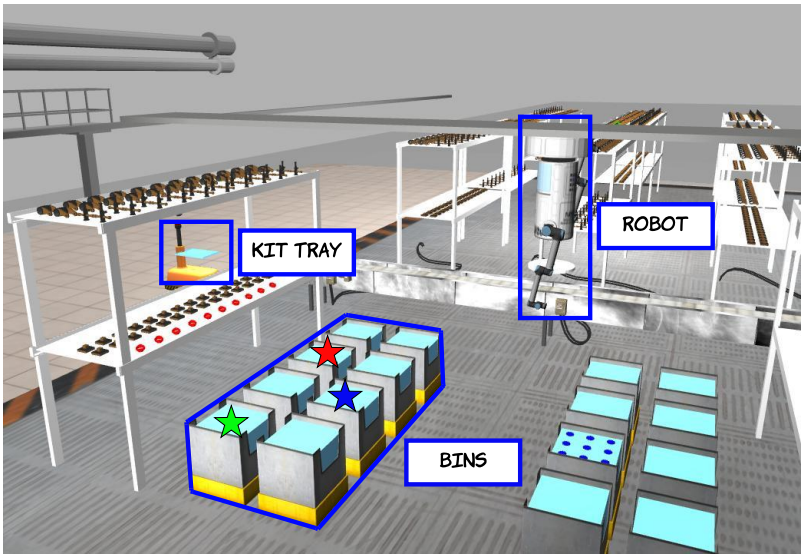

Figure 1: Snapshot of ARIAC 2020 Gazebo environment. Note: The bins depicted in this figure do not have any part but you will need to hard code how many parts each bin contains. Although not very relevant for the assignment we can expect the bin with the red star will only contain red parts, the bin with the green star will only contain green parts, and the bin with the blue star will only contain blue parts.

Figure 2: Available part types and colors in the bins.

2.3 Gripper Status

Although the robot has 2 grippers (1 mounted on each arm), we will use only 1 arm and 1 gripper.

The gripper has 2 states:

- Empty: In the initial state the gripper is always empty.
- Not empty: The gripper is holding a part.

2.4 Kit Tray Status

A kit tray is complete when it contains all the parts specified in the goal state.

The kit tray has 2 states:

- Complete: Kit tray has all the parts described in the goal state.
- Incomplete: Kit tray does not have all the parts described in the goal state. In the initial state the kit tray is empty, thus incomplete.

3 Goal State

The goal state is the state that you need to reach by performing a series of robot actions. For this assignment, the goal is simply specified by the number of parts to place in the kit tray. You need to prompt the user for the number of parts of each color to place in the tray.

```
How many red parts to place in the kit tray?  3
How many green parts to place in the kit tray?  2
How many blue parts to place in the kit tray?  5
```

In this example, the goal state is reached when the kit tray has 3 **red** parts, 2 **green** parts, and 5 **blue** parts.

4 Robot Actions

To go from the initial state to the goal state you will use robot actions. Each action of the robot will change the state of the environment and your program will stop when the goal state is reached.

In other words, use robot actions to go from 0 **red** part, 0 **green** part, and 0 **blue** part in the kit tray to 3 **red** parts, 2 **green** parts, and 5 **blue** parts in the kit tray (based on the example for the goal state I provided earlier).

The robot can perform only 2 actions: **pick** and **place**. Each action has a set of preconditions and a set of effects. All preconditions must be true before the action can be carried out. After executing the actions the effects are applied.

4.1 Pick

This action only picks up a part from the bin.

- Preconditions:
  - The gripper is empty.
  - There is at least one part of the specified color in the bin.
  - The kit is incomplete.
- Effects:
  - The gripper is not empty (it's holding a part).
  - Decrement the number of parts in the bin.

4.2 Place

This action only places a part in the kit tray.

- Preconditions:
  - The gripper is not empty (it's holding a part).
  - The kit is incomplete.
- Effects:
  - The gripper is empty.
  - Increment the number of part of the specified color in the kit tray.

5 Situations

There are some situations that you need to handle:

- After the user enters the number of parts you need to check if there are enough parts of the specified colors in the bins.
  - If there are enough parts then start your program to generate a plan.
  - If there are not enough parts then output a message and prompt the user to enter a new number. For instance:

```
--Not enough red parts in the bin to build the kit
How many red parts to place in the kit tray?
```

- When the kit is complete then output:
  - The generated plan.
  - The number of parts of each color that were placed in the kit tray.
  - The number of remaining parts in each bin.
- Below is an example of the output from the program. Please note that you do not need to have the exact format or the same sentences. However you must follow the instructions.

```
==========================================
There are 10 red parts in the bin.
There are 10 green parts in the bin.
There are 10 blue parts in the bin.
==========================================
How many red parts to place in the kit tray? 1
How many green parts to place in the kit tray? 2
How many blue parts to place in the kit tray? 2
==========================================
Generating a plan...

====================================
---Pick red part
---Place red part
====================================
---Pick green part
---Place green part
------------------------
---Pick green part
---Place green part
====================================
---Pick blue part
---Place blue part
------------------------
---Pick blue part
---Place blue part
====================================
Summary:
The kit tray has 1 red part(s) -- The bin has 9 red part(s) left
The kit tray has 2 green part(s) -- The bin has 8 green part(s) left
The kit tray has 2 blue part(s) -- The bin has 8 blue part(s) left
```

- Below is an example of the output from the program where the user entered more parts than there are in the bins.

```
==========================================
There are 10 red parts in the bin.
There are 10 green parts in the bin.
There are 10 blue parts in the bin.
==========================================
How many red parts to place in the kit tray? 12
--Not enough red parts in the bin to complete the kit
How many red parts to place in the kit tray? 1
How many green parts to place in the kit tray? 20
--Not enough green parts in the bin to complete the kit
How many green parts to place in the kit tray? 2
How many blue parts to place in the kit tray? 2
==========================================
Generating a plan...

======================================
---Pick red part
---Place red part
======================================
---Pick green part
---Place green part
----------------------
---Pick green part
---Place green part
======================================
---Pick blue part
---Place blue part
----------------------
---Pick blue part
---Place blue part
======================================
Summary:
The kit tray has 1 red part(s) -- The bin has 9 red part(s) left
The kit tray has 2 green part(s) -- The bin has 8 green part(s) left
The kit tray has 2 blue part(s) -- The bin has 8 blue part(s) left
```

6 Instructions

Grading:

- You need to use procedural programming and at least 2 functions: **pick** and **place**.
- You can write the whole program in `main.cpp`.
- Your program must follow the Google C++ style guide.
- Your program must be documented (Doxygen) and commented (C++ comments).
- After you run your program, you should get the following results in your terminal.

```
==========================================
There are 10 red parts in the bin.
There are 10 green parts in the bin.
There are 10 blue parts in the bin.
==========================================
How many red parts to place in the kit tray?
```

7 Submission

- Create a project rwa1-<lastname> and write your code in it.
- Include your **Doxyfile** in your project.
  - Do not include html files: We will be able to generate the documentation on our side.
- Zip your project and upload **rwa1-lastname.zip** to Canvas.