Exercises 1-3: Manipulation Planning (ex.py)

1. Uncomment run_ex1 in the main program. The display shows a robot and a cylindrical object on a table. When you type ‘l’/’r’ on the keyboard, the program will call the function planTransit to try to plan a path from the current configuration to grasp the object using the robot’s left/right hand. If successful, it will show an animation of the planned path.

   The planning process is as follows: (1) find a collision-free grasping configuration that satisfies the grasp’s IK constraint, (2) plan a collision-free path to the grasping configuration. However, it currently fails (for both hands) because it is unable to find a feasible solution in step (1). The reason why is that it runs the IK solver only starting from the current configuration, which may not yield a feasible IK solution.

   Describe and implement a reliable method for finding a grasping configuration. Your method should work reliably for grasping with the left hand.

   Discuss what is happening with the right hand. Why can’t your method find a solution?

2. Uncomment run_ex2 in the main program. The robot is now starting out with the object in its hand. When you type ‘f’ the robot will try to move the object forward by 15cm, and when you type ‘r’ the robot will try to move the object rightward by 15cm. The planTransfer function is called in order to do so via the following process: (1) find a collision-free ungrasping configuration that satisfies the placement’s IK constraint, and (2) plan a collision-free transfer path to the ungrasping configuration assuming the object is attached rigidly to the robot’s hand. But, this method is incomplete.
   a. First of all, the feasible() method of the TransferCSpace class is in error, because it incorrectly reports a collision between the hand and the object, and it fails to check object-terrain collisions. Fix this method.
   b. Second, key lines in (1) and (2) are missing. Implement these, consulting the implementation in Problem 1.
   c. Discuss why the forward transfer ‘f’ fails while the rightward transfer ‘r’ succeeds.

3. Uncomment run_ex3 in the main program. There are now 4 target locations on the table, ‘a’, ‘b’, ‘c’, and ‘d’, with the object starting at ‘a’. When one of these keys is typed, the robot should plan an entire motion from the current state that moves the object to the pressed target location. If you correctly implemented questions 1 and 2, typing ‘b’ should successfully move the object to location ‘b’. After one pick-and-place action you may type ‘n’ to stop animating the solution and then proceed to the next step. If you then type ‘a’, the robot should successfully move the object back to location ‘a’.

   a. The robot cannot reach locations ‘c’ and ‘d’. One possible solution is to place the object into a temporary location and then switch hands. Implement a new planner that enables you to move the object to locations 'c' and 'd' and then back to 'a'. (You may choose intermediate steps by hand.)

b. If your planner incorrectly picks a hand that can reach the current object's location but *cannot* reach the destination, the planner will spend a lot of time planning the transit before it discovers that this choice is infeasible. Describe how you might change the planner to discover the infeasibility faster. Give pseudocode (you may use subroutines like find_pregrasp_config, plan_transit_path, find_ungrasp_config, and plan_transfer_path).

c. It is still somewhat disappointing to have to instruct the robot through each intermediate step. Rather, it would be much more convenient to ask "Robot, please move the object to position Y" and have the robot figure out the intermediate steps automatically. Describe how you might design a multi-modal planner to do so. Describe the mode families, mode parameters, and mode coparameters.