Explanation:

My program implements GOAP using a world-state vector which has only two elements: “inTree” and “foodCount”. Both of these have a corresponding “int” value, where “inTree” can have values 0 (representing not being in a tree) and 1 (representing being in a tree). “foodCount” can have values 0, 1, 2, or 3, which represents the amount of food they are carrying. In the assignment it is said squirrels can’t mix garbage and nuts, so my solution to this was to merge nuts and garbage into “food”, where garbage is worth 3 points of food, and each nut is worth just one. In this way, squirrels can only pick up garbage if they have no nuts.

Here is the list of actions, along with a short description and their pre/post conditions:

- **IdleSitAction**: Squirrel simply sits there.
  Precondition: inTree = 0
  Postcondition: N/A
- **IdleAction**: The squirrel wanders to a random location within a short range.
  Precondition: inTree = 0
  Postcondition: N/A (though, squirrel will have moved)
- **ClimbTreeAction**: The squirrel goes into a tree.
  Precondition: inTree = 0
  Postcondition: inTree = 1
- **StoreFoodAction**: The squirrel stores its food in the tree.
  Preconditions: inTree = 1, foodCount != 0
  Postcondition: foodCount = 0
- **DescendTreeAction**: The squirrel gets off the tree it is currently occupying.
  Precondition: inTree = 1
  Postcondition: inTree = 0
- **TakeNutAction:** The squirrel picks up a nut from the ground and adds it to its food count.
  Precondition: inTree = 0, foodCount != 3
  Postcondition: foodCount++
- **InvestigateGarbageAction**: The squirrel attempts to get food from a garbage can.
  Preconditions: inTree = 0, foodCount = 0
  Postconditions: foodCount = 3 (if there was indeed junk in the can. No state-vector change happens otherwise)

Finally, here is one example path for each of the required squirrel behaviors:

- **“At least two idle behaviors, one of which involves roaming/exploring randomly”**: If in a tree when planning, the squirrel receives a “DescendTreeAction” and then randomly decided between a “IdleSitAction” or an “IdleAction” (which is the random exploring). If not in a tree, simply gets the “IdleSit” or “Idle” Action on their own.
- **“The ability to pick up nuts and bring them back to their home tree [...]”:** If the squirrel is not in a tree, an example of path would be TakeNutAction -> TakeNutAction -> TakeNutAction -> ClimbTreeAction -> StoreFoodAction.
  In the case where a squirrel may start in a tree with two nuts when gaining a plan, it could go DescendTreeAction -> TakeNutAction -> ClimbTreeAction -> StoreFoodAction. Many other similar possibilities exist.
  (Note: Both these example plans assume the squirrel has the required amount of nuts in memory. Otherwise the plans would fail since TakeNutAction could not be executed, see more details on how my GOAP works in the code)
- **“They sometimes gather food from garbage cans [...]”:** If a squirrel is on the ground with no food and a nearby garbage can is full, an example plan would be InvestigateGarbageAction -> ClimbTreeAction -> StoreFoodAction.
  When a squirrel gets to a garbage can, however, you can see in the code (LateUpdate in SquirrelScript + this action in the Action script) that there is a verification when performing the action that the garbage is full. If it is not full, the squirrel instead gets stuck in the can for two seconds, halting any actions, which then forces the squirrel to abandon its plan and replan afterwards.
- **“If the Player comes too close (choose a reasonable distance) they immediately flee, seeking refuge in a tree until the player is further away. No more than one squirrel can be in a given tree at the same time”:** If a squirrel is in a tree when a player approaches, it will cancel its current plan and stop making new plans, remaining there until the Player leaves. If the squirrel is somewhere on the ground, an existing plan is dropped and a new plan consisting of a single ClimbTreeAction is given, with a target corresponding to the nearest tree not containing another squirrel (see how I implemented targets into my actions in the code. Basically, each Action has a target, and the action is not executed until the Squirrel has successfully used pathfinding to reach the target. As soon as a target is reached, the action is executed, so in this case, the squirrel climbs the tree as soon as it arrives.)