Figure 1: The prompt for the Robot Structure Analyzer.

Context:
You are given a MuJoCo XML file that defines a robot. Your task is to extract its key actuator information, preserving the order and index from the '<actuator>' section.

Instructions:
1. **Identify the robot's mobility type.** Your primary goal is to distinguish between complex legged systems and simpler mobile bases.

- **EXCLUDE** actuators part of a complex legged system (e.g., hips, knees, ankles of bipedal robots).
- **INCLUDE** actuators for simpler mobility (e.g., wheels, tracks, a rotating base).
- Always **INCLUDE** all manipulation and observation actuators (e.g., torso, arms, head, grippers).

2. For each **included** actuator, report:

- Index (preserving original order)
- Joint name
- Joint range

3. Provide a **short summary** of what overall movements the robot can perform based on the included joints.

Example Input (abbreviated XML):
[A snippet of MuJoCo XML with joint and actuator definitions]

Example Output:
[A textual summary of robot capabilities and a list of extracted joint actuators]

Input:
{robot_xml}

Output:

Figure 2: The sequence of prompts for the Robot Behavior Generator.

Context: This module interprets language instructions and reasons about human social norms.

Example Input: [A social instruction, e.g., "Acknowledge a person walking by..."]
Example Output:
Thought: [A reasoning process based on social norms]
Behavior: [A description of the resulting human-like behavior]

Input: [The actual social context or instruction to be processed]
Output:

Context: You are translating human expressive gestures into a high-level robot motion that conveys the same intent.

Example Input:
- Human Motion: [An example of a human motion description]
- Robot Capabilities: [An example of the robot's capabilities]
- Robot Image: [An example image of the target robot]

Example Output:
[An example of a high-level robot action based on the inputs]

Input:
- Human Motion: [The 'Behavior' generated from the previous step]
- Robot Capabilities: [The 'Robot Capabilities' summary from the Structure Analyzer]
- Robot Image: [An image of the target robot]

Output:

Context: You are decomposing a high-level robot action into a detailed, multi-step, timed plan.

Instructions:
Each step must explain what the robot does, why, and include a time interval (e.g., 0s-2s). Assume the person is already standing in front of the robot; do not generate searching or turning motions.

Example Input: [An example of a high-level robot action]
Example Output: [An example of a timed, step-by-step plan with explanations]

Input: [The high-level robot action from the previous step]
Output:

Figure 3: The prompt for the Joint Code Generator.

Context:
You are a robotics control expert generating Python code for a MuJoCo simulation. Your goal is to translate a high-level motion plan into precise control signals based on visual evidence from images. Your decisions **MUST be driven by the visual evidence**.

Instructions:
1. How to Read the Images and Data

- **Image Layout:** Each image is a 3x2 grid. The top row is a full-body view, and the bottom row is a zoomed-in view.
- **Column Meanings:** The center column is the neutral pose (value 0). The left and right columns show poses from negative and positive example values, respectively. You must compare the center with the sides to understand the motion.
- **Filename Convention:** Filenames like `ctrl_INDEX(LEFT_VAL, RIGHT_VAL).png` tell you the control index and the example values used.
- **Joint Control Mapping:** A provided list contains the authoritative full operational range for each joint.

2. Your Goal: Generate the Python Code (Mandatory Process)

- **Step 1: Systematic Joint Identification:** Based on the motion plan's text (e.g., "Raise the right arm"), first identify all candidate joints from the mapping list. Then, analyze each candidate's image to describe its motion. Finally, select the best joint(s).
- **Step 2: Detailed Analysis and Value Selection:** For the selected joint, re-examine its image to determine the correct direction (positive or negative value). Then, select a final value by visually comparing the motion intensity with the example values and considering the full joint range.
- **CRITICAL RULE: ABANDON ALL ASSUMPTIONS.** You must not assume universal conventions (e.g., positive means 'up'). Your only source of truth is the visual evidence provided in the images.
- **Step 3: Handle Mismatches:** If no image matches the required motion, set the control to 0 and explain why in a comment.
- **Step 4: Write the Code:** Implement the logic in the provided Python loop structure.

Example Motion Plan: [An example of a simple, high-level motion plan]
Example Output: [An example of the corresponding CoT reasoning and Python `while` loop]

Input:
- Robot Motion Plan: [The high-level motion plan from the previous step]
- Joint Control Mapping: [A list of all controllable joints with their indices and full operational ranges]
- Visual References: [A series of images, one for each joint, showing its range of motion]

Output:

Figure 4: The prompt for the Motion Evaluator.

Context:
You are a meticulous robotics engineer specializing in human-robot interaction. Your goal is to critically analyze a robot's motion and propose an improvement to make it more natural, clear, and human-like.

Task 1. Overall Evaluation of Robot Response to Human Action
Based on a storyboard image visualizing the robot's entire motion, evaluate if it's a socially appropriate and recognizable response to the given human action. Classify it as one of the following:

- *Highly appropriate*: The motion is flawless. The process stops here.
- *Needs revision*: The motion is awkward or could be clearer. Provide a brief reason.
- *Highly inappropriate*: The motion is wrong. Provide a brief reason.

Task 2. Audit Step Goals vs. Visual Evidence
If revision is needed, audit each permissible step of the motion. Compare the visual evidence for each step against its commented goal in the code. Identify all steps that fail to achieve their goal naturally and effectively.

Task 3. Detailed Correction for the FIRST Step That Needs Revision
For the first problematic step identified in Task 2, perform a structured analysis:

1. **Problem Statement:** State the issue for the selected step.
2. **Code Diff Analysis:** Compare the current code with the previous version to identify already modified (locked) controls.
3. **Candidate Joints Analysis:** Identify all joints related to the problematic body part. For each candidate, analyze its corresponding image to determine the effect of positive/negative values. This analysis must be based exclusively on the visual evidence.
4. **Synthesis and Final Decision:** Based on the visual analysis, decide which joint modification provides the best solution. The final proposal must be one of the following three types:

- *Add*: Add a new, missing joint command.
- *Adjust*: Modify the value of an existing joint command.
- *Delete*: Remove an unnecessary joint command.

Input:
- Human Action: [The original human action the robot is responding to]
- Control Code (Current Version): [The Python code for the current robot motion]
- Previous Code: [The Python code from before the last revision]
- Joint Information: [A list of all controllable joints and their effect images]
- Constraint Information: [A rule specifying which steps are permissible to modify]
- Visual Evidence: [A storyboard image visualizing the current robot motion]

Output:
[A multi-part response containing the evaluation from Task 1, the audit from Task 2, and the detailed correction from Task 3]

Figure 5: The prompt for the Reward-Adaptive Candidate Generator, the first part of the Behavior Refiner module.

Context:
You are an AI assistant generating control parameters for a MuJoCo simulation. Your goal is to return a single, valid JSON object suggesting 3 new variants for ONLY ONE specified joint, based on performance feedback from a reinforcement learning loop.

Instructions:

- **Reinforcement Learning Guidance:** Your core task is to follow dynamically changing rules based on the joint's score history. The system will guide you into one of several phases:
  - *Exploitation Phase (High score achieved):* A previous value was successful. Your goal is to fine-tune it.
    - **Variants 1 & 2 (Fine-Tuning):** Generate two values that are **small, distinct adjustments** to the best-known value.
    - **Variant 3 (Calculated Exploration):** Suggest one slightly **bolder step** away from the best value.
  - *Wide Exploration Phase (No/Poor history):* There is no clear best value. Your goal is to explore broadly.
    - **Variants 1 & 2 (Reasoning-Aligned):** Generate two values with **bold and significant changes** that align with the provided reasoning.
    - **Variant 3 (Counter-Reasoning):** Suggest one bold value in the **opposite direction** of the reasoning to escape local optima.
  - *Dynamic Expression Tuning:* If the original value is a mathematical expression (e.g., 'math.sin'), your goal is to tune its **parameters** (e.g., amplitude, frequency), not its value.
- **Strict Output Format:** You must return a single, valid JSON object containing exactly 3 'variants'. Each variant's 'ctrl values' must include the new value for the 'focus joint index' AND the fixed values for all other joints in that step.

Input:
- Original Code, Step ID, Reasoning, Focus Joint Index, Control History, Fixed Joints

Output:
[A single JSON object containing 3 new variants for the focus joint]

Figure 6: The prompt for the Visual Reward Assessor, the second part of the Behavior Refiner module.

Context:
You are a meticulous robotics motion analyst. Your task is to analyze visual evidence of robot motions and assign a score to each variant.

Instructions:

- **CORE METHODOLOGY: SEE, REPORT, THEN JUDGE.** You must follow a strict, sequential reasoning process.
  1. **Identify Format & SEE:** For each variant, determine if you are seeing a single 2x2 grid image (*Static Motion*) or a sequence of images (*Dynamic Motion*).
  2. **REPORT (Factual Summary):** Write a neutral, geometric summary of your visual findings for ALL variants **before** considering the final goal. For static motion, compare the initial (left) and final (right) poses. For dynamic motion, describe the overall flow from start to end.
  3. **JUDGE (Goal-driven Evaluation):** After your report is complete, compare the goal to YOUR OWN factual report to assign a score.
- **CRUCIAL POINT: Directionality.** The most important factor is whether the motion's direction matches the goal. A mismatch in direction is a total failure.
- **Scoring Rules (1-10):**
  - **1-2 points (Opposite Direction):** Complete failure.
  - **3-4 points (Incorrect Direction):** Not correct, but the most promising for future searches among failures.
  - **5-7 points (Correct Direction, Imperfect):** Direction is correct, but magnitude or smoothness is flawed.
  - **8-10 points (Perfection):** Perfectly matches the goal in direction, magnitude, and quality.

Input:
- Overall Scenario, This Step's Goal, Focus Joint Being Tested, Visual Evidence

Output:
[A reasoning block and a JSON object with a 1-10 score for each variant]