

Behavioral Contracts

1. Operation: `moveWorker(fromX, fromY, toX, toY)`

Preconditions:

1. Worker exists at the starting position (`fromX`, `fromY`):

- The specified starting coordinates must contain the worker that is being moved.
- *Example:* Worker 0 is currently at (0, 0), so `fromX = 0` and `fromY = 0`.

2. Worker belongs to the current player:

- The worker being moved must belong to the player whose turn it currently is.
- *Example:* Player A is attempting to move their own worker, not Player B's.

3. Valid Source Position:

- The starting position (`fromX`, `fromY`) must be within the 5x5 grid.
- *Example:* (`fromX`, `fromY`) must satisfy $0 \leq \text{fromX} < 5$ and $0 \leq \text{fromY} < 5$.

4. Valid Destination Position (`toX`, `toY`):

- The destination must be within the 5x5 grid and adjacent to the worker's current position.
- *Example:* If the worker is at (0, 0), they can move to (0, 1) or (1, 0).

5. Height Restriction:

- The worker can only move to a position where the tower height difference between the current and destination positions is ≤ 1 .
- *Example:* The worker can move from a level-1 tower to a neighboring position at level 0, 1, or 2, but not to level 3.

6. Destination is unoccupied:

- The destination position must not contain another worker or a dome.
- *Example:* The worker cannot move to (1, 1) if it's occupied by another worker or a level-3 tower with a dome.

7. The tower is complete

- The grid has been initialized with positions for all towers and empty fields.
- Each position on the grid can have a tower with 0 to 3 levels.
- Only level-3 towers are allowed to have domes.
- Workers cannot occupy a position with a dome.
- *Example:*

```
function isTowerComplete(towerPosition, grid):
    tower = grid[towerPosition]
    if tower.towerLevel == 3 and tower.hasDome:
        return true // Tower is complete
    return false    // Tower is incomplete
```

Postconditions:

1. Worker's position is updated:

- The worker's new position is updated to the destination coordinates (**toX**, **toY**).
- *Example:* Worker 0 moves from (0, 0) to (0, 1), and the worker's new position is (0, 1).

2. Previous position is unoccupied:

- The worker's previous position (**fromX**, **fromY**) is now unoccupied.
- *Example:* Position (0, 0) is now empty after Worker 0 moves to (0, 1).

3. Board state is updated:

- The board's internal state is updated to reflect the worker's new position.
- *Example:* The board at (0, 1) is now marked as occupied by the worker.

4. Player's turn continues:

- The player can proceed to the building phase if the move is valid.
- *Example:* After a valid move, Player A proceeds to the building phase.

5. Invalid move does not change state:

- If the move is invalid, the worker's position remains unchanged.
- *Example:* If the destination is occupied, Worker 0 stays at (0, 0), and the player is prompted to try again.

2. Operation: **build(workerX, workerY, buildX, buildY)**

Preconditions:

1. Worker exists at the specified position (workerX, workerY):

- The worker must be present at the specified position.
- *Example:* Worker 0 is at (0, 1), so **workerX = 0** and **workerY = 1**.

2. Building position is adjacent to the worker:

- The building position (**buildX**, **buildY**) must be adjacent to the worker's current position.
- *Example:* Worker at (0, 1) can build at (0, 0), (0, 2), (1, 1), or (1, 0).

3. Building position is unoccupied:

- The building position must not contain another worker or a dome.
- *Example:* If the building position (0, 0) contains a dome or another worker, building is not allowed.

4. Building position is within bounds:

- The building position must be within the 5x5 grid.
- *Example:* (buildX, buildY) must satisfy $0 \leq \text{buildX} < 5$ and $0 \leq \text{buildY} < 5$.

5. Building is valid:

- The player can only build domes on level-3 towers, and can build blocks on towers below level 3.
- *Example:* A block can be built at a level-2 tower, while a dome is built on a level-3 tower.

Postconditions:

1. Block or Dome is built:

- If the building operation is valid, the height of the specified build position (buildX, buildY) is increased.
- *Example:* The tower at (0, 2) was at level 2, and after the build it increases to level 3.

2. Board state is updated:

- The board reflects the new tower height or presence of a dome at the build position.
- *Example:* The internal board state for position (0, 2) changes from level 2 to level 3 after building.

3. End of Turn:

- After a successful build, the player's turn ends, and the game passes to the next player.
- *Example:* After Player A builds, it's now Player B's turn.

4. Invalid build does not change state:

- If the build action is invalid, the game state remains unchanged.
- *Example:* If Player A tries to build on a dome, the game state remains the same and Player A must choose another build location.

3. Operation: checkVictory()

Preconditions:

1. Player has completed a move:

- The player must have moved a worker.
- *Example:* Player A just moved Worker 0 to (0, 3).

2. Worker's position is valid:

- The worker must be within the 5x5 grid.
- *Example:* Worker 0 is at (0, 3).

Postconditions:

1. **Victory detected:**

- If the worker is on a level-3 tower, the player wins the game.
- *Example:* Worker 0 reaches level 3 at position (0, 3), and Player A wins.

2. **No victory, continue game:**

- If the worker is not on a level-3 tower, no victory is detected, and the game continues.
- *Example:* Worker 0 moves to (0, 1), but since it is not on a level-3 tower, the game continues.