UNIVERSITY OF JOHANNESBURG

FACULTY OF SCIENCE

COMPUTER SCIENCE 1A **DESIGN**

Problem Description

Utopian Time Synch is developing a game based around solving weight based puzzels that involve transporting the correct number of standardized weights to specific scale tiles which will weight them. Each scale is set up to expect a precise amount of weight to be placed on it. If the correct amount of weight is placed on all scales then the game is won. If any scale has the incorrect amount of weight placed on it then the game ends in failure.

Input & Output;

Input	
Input Description	*Mechanism*
Number of scales, weights and size	Command-line arguments
Options for game play	Standard input
Gameplay moves	Standard input
Output	
Output Description	*Stream (optional)*
NewPosition/newWold Display	Standard Output
Errors	Standard Output

Data Format

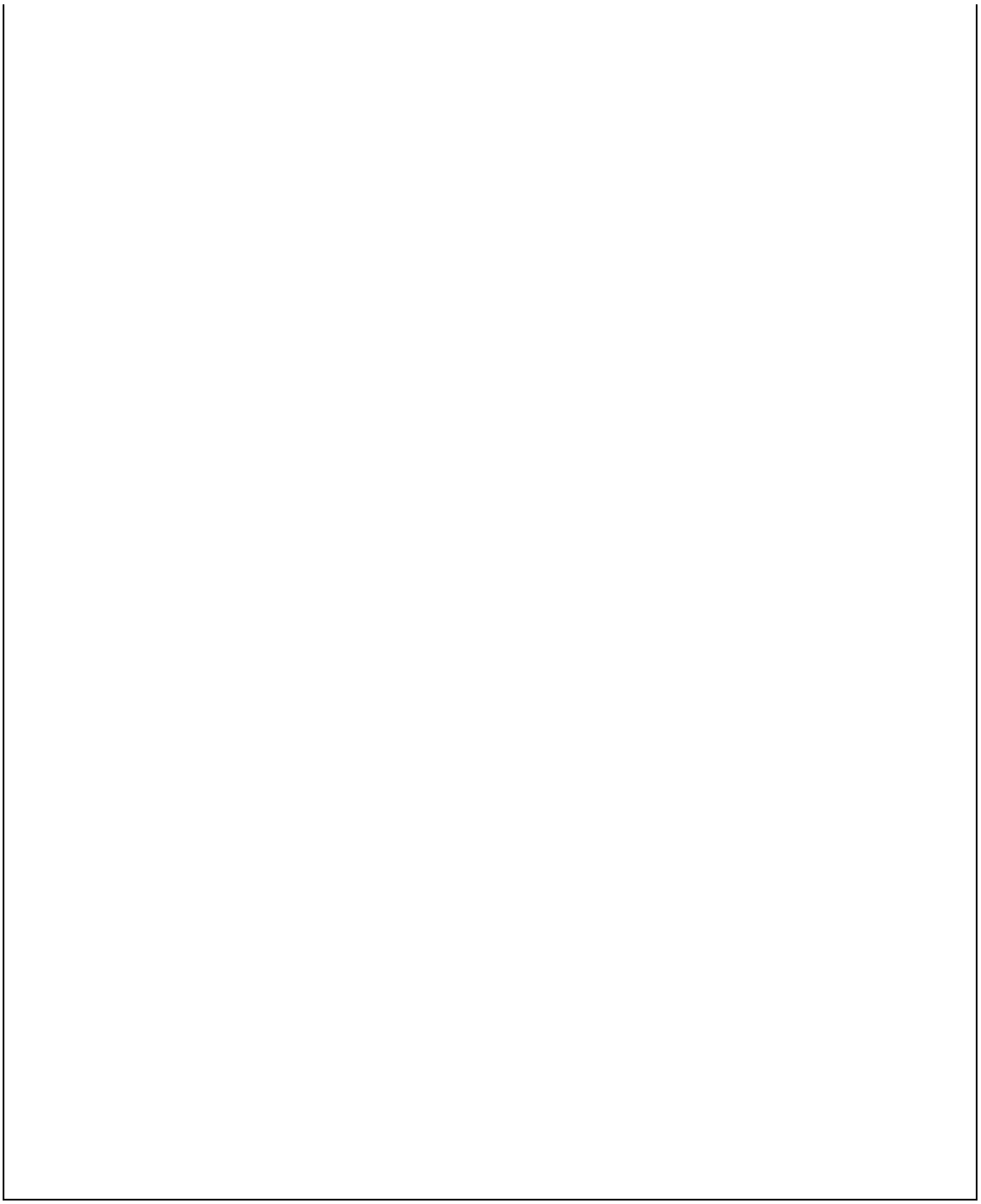
Identifier	*Data Type*	*Description*
envSize	integer	For setting environment size
totalWeights	integer	For taking number of weights in environment

numScales	integer	For taking the number of scales in environment

Pseudo Code

```
 // Define constants
const MAX_WEIGHT <= 10
const MAX_SCALE_VALUE <= 10
const MAX_ENV_SIZE <= 20

// Initialize environment function
function initializeEnvironment(envSize, totalWeights, numScales):
    Place player at center of environment
    Randomly place numScales scales in env with random values
    Distribute totalWeights as weights in env

// Move player function
function movePlayer(playerPos, newPos):
    if newPos is within bounds and is empty:
        Move player to newPos
        return true
    else:
        return false

//Drop weight function
function dropWeight(playerPos):
    Find empty neighboring position next to playerPos
    If found, place weight at that position
    If not found, return false

// Check game state function
function checkGameState():
    if any scale has a non-zero value:
        return ONGOING
    else if player is standing on a scale with incorrect weight:
        return FAILURE
    else:
        return VICTORY

// Main game loop
initializeEnvironment(envSize, totalWeights, numScales)
while game state is ONGOING:
    Display env
```

```
    Accept player movement input
    Move player accordingly
    Check if player drops weight
    Check game state
if game state is FAILURE:
    Display "Game Over! You failed to meet the conditions."
else if game state is VICTORY:
    Display "Congratulations! You've won the game!"
```



UML Activity Diagram