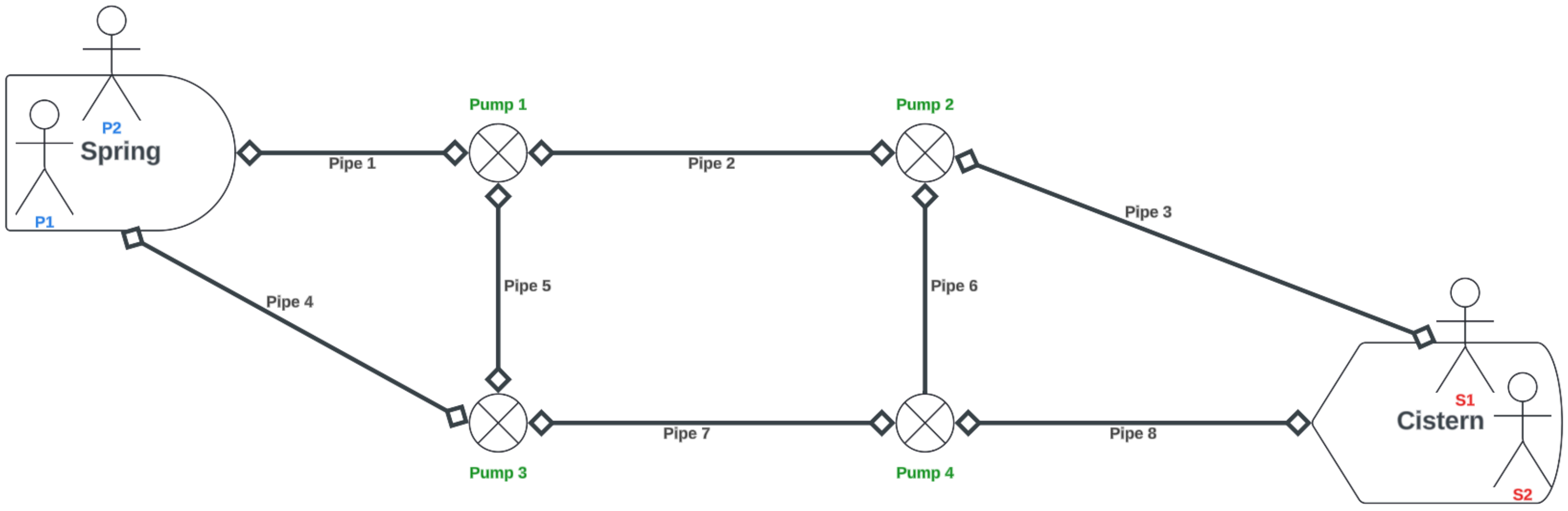
Testing Guide for the Prototype Program

Team Mansaf

The following assumptions shall be made in the testing procedure (important):

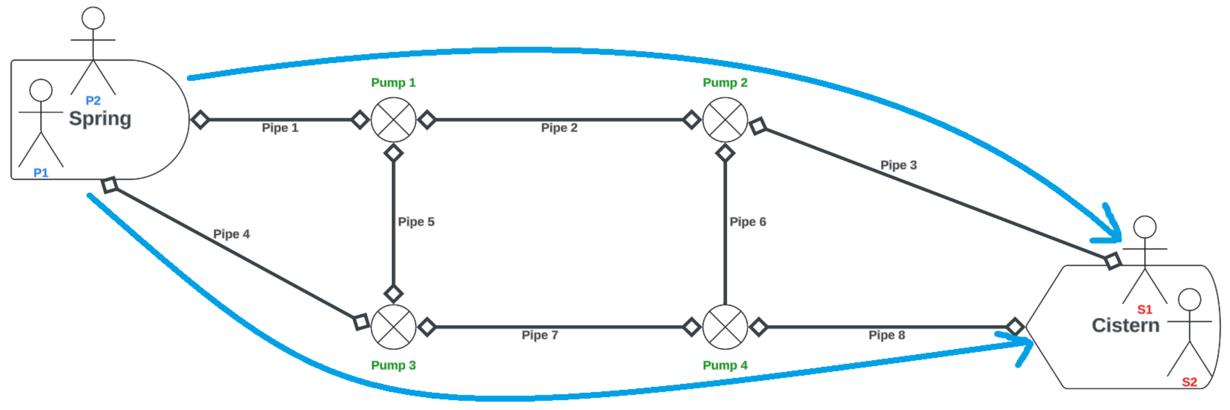
- All the tests will start from the map shown below:



- The diamonds at the endpoints of pipes represent pipe ends. Note that Pipe 6 does not have a pipe end connecting it to Pump 4, so there can be no water flow between them in this state (water will leak into the desert).

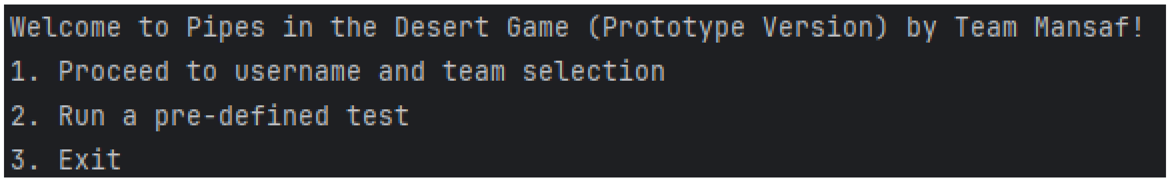
- The starting point of the Plumbers is the spring, and the starting point of the Saboteurs is the Cistern.

- There exists an initial water flow in the game, as follows:



- The turn order in the tests will be Plumber 1 -> Saboteur 1 -> Plumber 2 -> Saboteur 2.
- We have added supplementary diagrams to aid in visualizations for some tests.
- The output of the test files may not be word-by-word identical to the output in each test below but will have the same logic and ideas.
- The input and output data streams are somewhat simplified to avoid bloating the test case dialogue with commands that are irrelevant to the test case and keep it understandable. Since our game is turn-based, after a Saboteur makes a move, it will be a Plumber's turn, followed by the other Saboteur(s) and the other Plumber(s), meaning that after a Saboteur plays a turn, there will be a minimum of 3 other turns before they can do something else. This is somewhat inconvenient for the test cases that involve players moving continuously. Let's say a Plumber wants to get from the Spring to the Cistern in our above diagram, they would have to go from the Spring to Pipe 1, from Pipe 1 to Pump 1, from Pump 1 to Pipe 2, and so on. Following each of these movements there are 3 other turns that will be somewhat irrelevant to the test case, so we have decided to take such shortcuts:
  "Plumber 1 moves from the Spring to Pipe 1, all other players pass", etc.
- Go through the deployment guide in **Chapter 10. Prototype** document to grasp the correct way to execute the program.

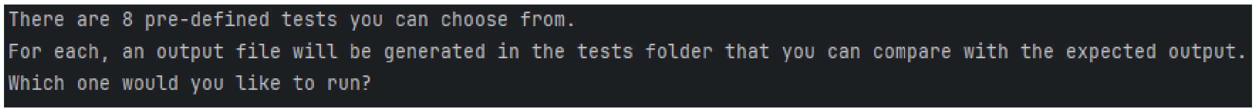
The way to execute the tests is to run the program from the Main class, after which you will be greeted with the starting menu:



```
Welcome to Pipes in the Desert Game (Prototype Version) by Team Mansaf!
1. Proceed to username and team selection
2. Run a pre-defined test
3. Exit
```

From this starting menu, choose **2. Run a pre-defined test**

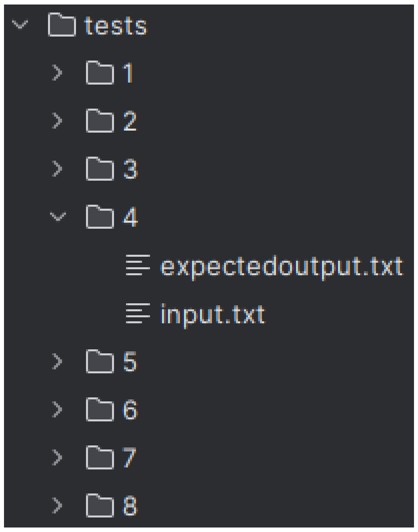
You will then receive the below message:



```
There are 8 pre-defined tests you can choose from.
For each, an output file will be generated in the tests folder that you can compare with the expected output.
Which one would you like to run?
```

Then enter an integer corresponding to the test case you'd like to run (1-8). 8.2.1 in this document corresponds to test number 1, 8.2.2 corresponds to test number 2, and so on.

After running the test, you will most likely be greeted with a *NoSuchElement* exception, which indicates that there are no more lines left to read (the whole input.txt file was read), at which point you should stop the main program and check the newly generated output.txt. You can go to the tests folder, which will have subfolders for each test, each containing an input.txt and an expectedoutput.txt.



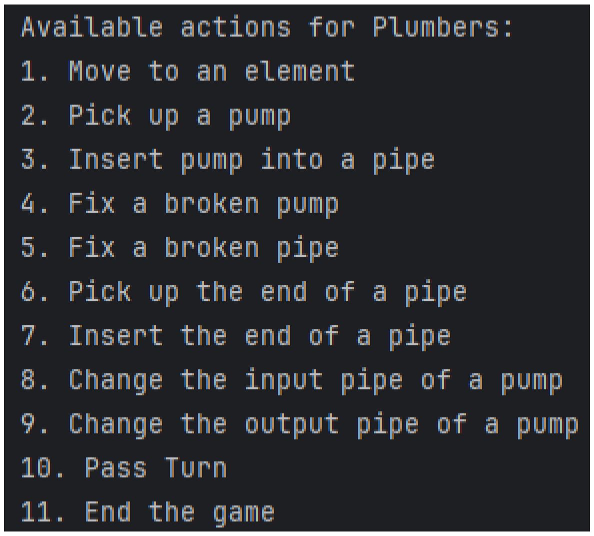
```
tests
  1
  2
  3
  4
    expectedoutput.txt
    input.txt
  5
  6
  7
  8
```

As mentioned previously, based on your selection, an output.txt will be generated too, which you can compare with the expectedoutput.txt.

- The first phase of "input" for the test cases will be initializing the game with 4 players named Plumber 1, Plumber 2, Saboteur 1, and Saboteur 2. This corresponds to the following input:
  4
  Plumber 1
  1
  Plumber 2
  1
  Saboteur 1
  Saboteur 2

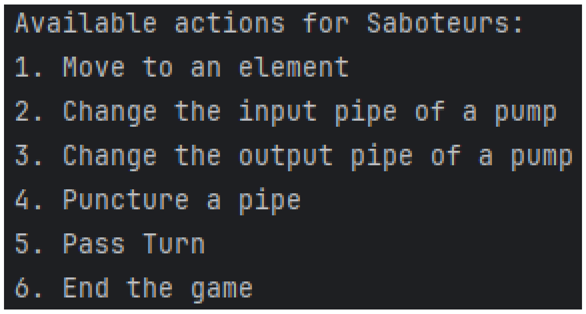
- Each number in the input.txts corresponds to an action (and sometimes, an element).

For Plumbers:



```
Available actions for Plumbers:
1. Move to an element
2. Pick up a pump
3. Insert pump into a pipe
4. Fix a broken pump
5. Fix a broken pipe
6. Pick up the end of a pipe
7. Insert the end of a pipe
8. Change the input pipe of a pump
9. Change the output pipe of a pump
10. Pass Turn
11. End the game
```

For Saboteurs:



```
Available actions for Saboteurs:
1. Move to an element
2. Change the input pipe of a pump
3. Change the output pipe of a pump
4. Puncture a pipe
5. Pass Turn
6. End the game
```

8.2.1 Plumber changing the output pipe of a pump, and fixing a pipe that was punctured by the Saboteur.

- **Description**

A saboteur will move from their starting position, the cistern, to Pipe 2 and puncture it. The plumber would want to fix it but will not initially be able to do so, as Pipe 2 is occupied by the Saboteur who punctured it. Therefore, the plumber changes the output pipe of Pump 1 from Pipe 2 to Pipe 5, to minimize water leaks. Knowing this, the Saboteur moves downwards. Finally, the plumber advances to Pipe 2 and fixes it.

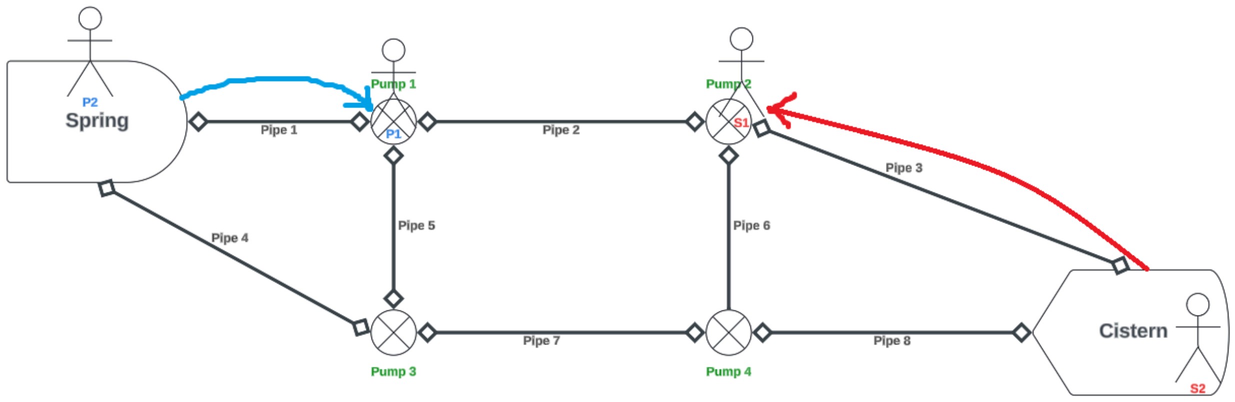


Fig 1. Saboteur 1 moves to Pump 2, and Plumber 1 moves to Pump 1.

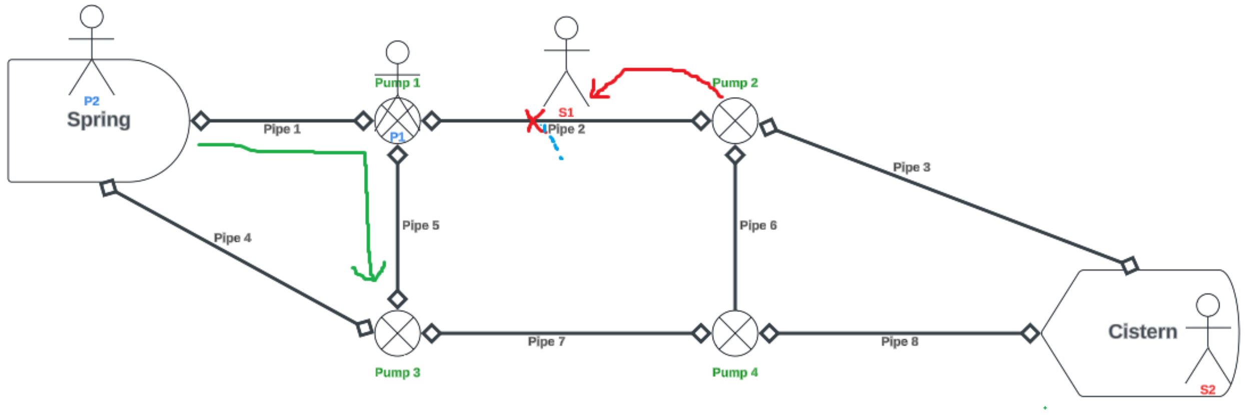


Fig 2. Saboteur 1 moves to Pipe 2 and punctures it, Plumber 1 changes the output pipe of Pump 1 to Pipe 5.

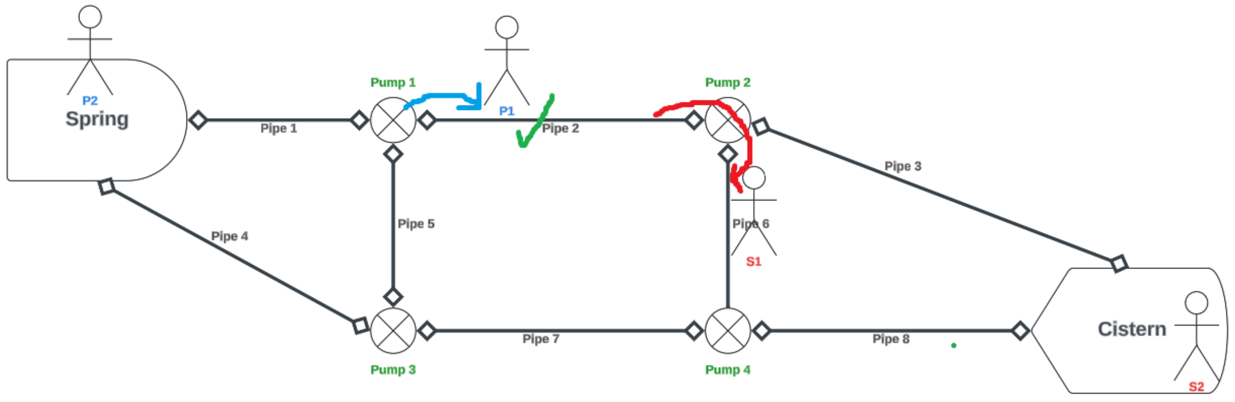


Fig 3. Saboteur 1 moves to Pipe 6, and Plumber 1 moves to Pipe 2 and fixes it.

- **Unit of functionality to be tested, possible failures**

The move(Element e) method which both players have.
The puncture(Pipe p1) method which the saboteur has.
The changeOutputPipe(Pump p1, Pipe p2) method.
The fixPipe(Pipe p) method of the Plumber.

Possible failures include the pipe not being punctured/fixed even after the players issue the commands.

- **Input**

4
Plumber 1
1
Plumber 2
1
Saboteur 1
Saboteur 2

Plumber 1: 1 (corresponds to “Move to an element”)
Pipe1
1
Pump1

Saboteur 1: 1
Pipe3
1
Pump2

– All other players pass their turns –

Saboteur 1: 1
Pipe2
4

– All other players pass their turns –

Plumber 1: 9
2 (corresponds to “Pipe 5”)
10

Saboteur 1: 1
Pump2
1
Pipe6

– All other players pass their turns –

Plumber 1: 1
Pipe2
5

- **Output**

Proceeding to the number of participating players menu.

4 players will participate in the game.

Proceeding to the username and team selection menu.

The name "Plumber 1" is validated.

Plumber 1 chose the Plumbers team.

The name "Plumber 2" is validated.

Plumber 2 chose the Plumbers team.

The name "Saboteur 1" is validated.

Saboteur 1 was automatically placed in the Saboteurs team.

The name "Saboteur 2" is validated.

Saboteur 2 was automatically placed in the Saboteurs team.

Initializing the game...

The game’s elements have been initialized successfully.

The game and timer have started!

Plumber 1 moved to Pipe1

Plumber 1 moved to Pump1

Saboteur 1 moved to Pipe3

Saboteur 1 moved to Pump2

Plumber 2 passed their turn.

Saboteur 2 passed their turn.

Plumber 1 passed their turn.

Saboteur 1 moved to Pipe2

Saboteur 1 punctured Pipe2

Plumber 2 passed their turn.

Saboteur 2 passed their turn.

Plumber 1 changed the output pipe of Pump1 to Pipe5

Plumber 1 passed their turn.

Saboteur 1 moved to Pump2

Saboteur 1 moved to Pipe6

Plumber 2 passed their turn.

Saboteur 2 passed their turn.

Plumber 1 moved to Pipe2

Plumber 1 fixed Pipe2

8.2.2 Plumber picks up a randomly generated pump at the cistern and inserts it elsewhere.

- **Description**

Plumber 1 will move to the cistern from the spring to pick up a pump that was randomly generated at the cistern. They will then move to Pipe 6 and insert the new pump they have acquired, thus adding the pump to the system and splitting Pipe 6 into two pipes.

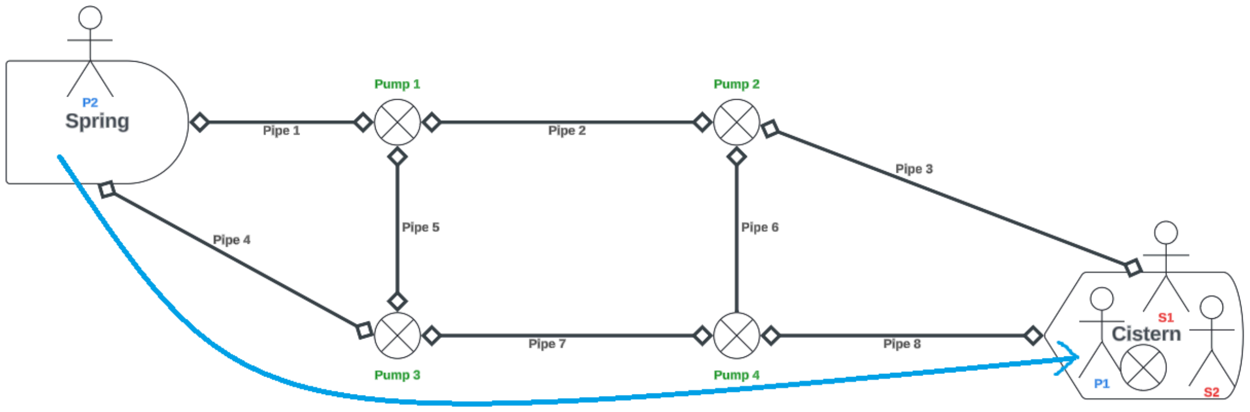


Fig 1. Plumber 1 goes to the cistern and picks up a pump.

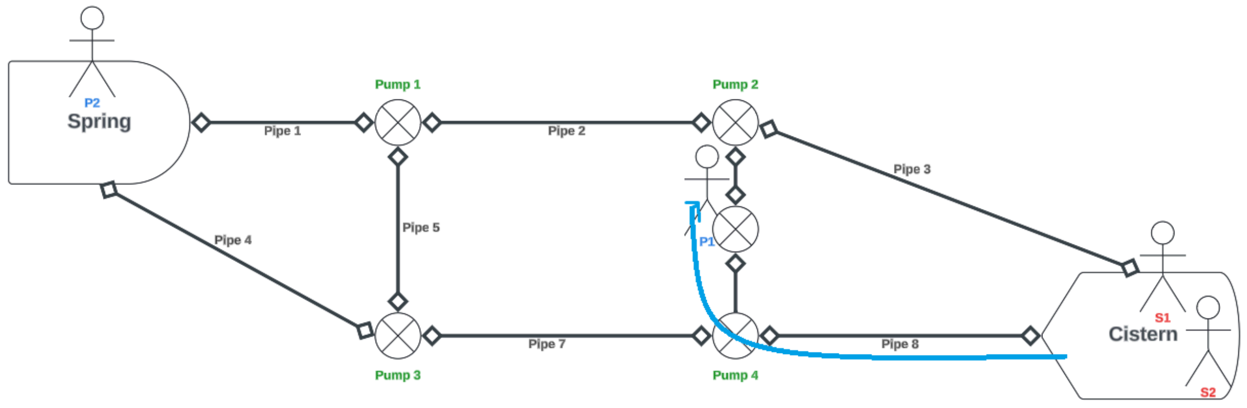


Fig 2. Plumber 1 moves to Pipe 6 and inserts a new pump, splitting the current pipe into two pipes.

- **Unit of functionality to be tested, possible failures**

The move(Element e) method which both players have.
The random pump generation at the Cistern.
The getPump(Pump p) method of the Plumber.
The insertPump(Pump pump, Pipe pipe1, Game g1) method of the Plumber.

Possible failures include the pump not being picked up/inserted even after the player issues the commands.

- **Input**

4
Plumber 1
1
Plumber 2
1
Saboteur 1
Saboteur 2

Plumber 1: 1
Pipe4
1
Pump3

– All other players pass their turns –

Plumber 1: 1
Pipe7

1
Pump4

– *All other players pass their turns* –

Plumber 1: 1
Pipe8
1
Cistern

– *All other players pass their turns* –

Plumber 1: 2
1
Pipe8

– *All other players pass their turns* –

Plumber 1: 1
Pump4
1
Pipe6

– *All other players pass their turns* –

Plumber 1: 3

- **Output**

Proceeding to the number of participating players menu.

4 players will participate in the game.

Proceeding to the username and team selection menu.

The name "Plumber 1" is validated.

Plumber 1 chose the Plumbers team.

The name "Plumber 2" is validated.

Plumber 2 chose the Plumbers team.

The name "Saboteur 1" is validated.

Saboteur 1 was automatically placed in the Saboteurs team.

```
The name "Saboteur 2" is validated.

Saboteur 2 was automatically placed in the Saboteurs team.

Initializing the game...

The game’s elements have been initialized successfully.

The game and timer have started!

Plumber 1 moved to Pipe4

Plumber 1 moved to Pump3

Saboteur 1 passed their turn.

Plumber 2 passed their turn.

Saboteur 2 passed their turn.

Plumber 1 moved to Pipe7

Plumber 1 moved to Pump4

Saboteur 1 passed their turn.

Plumber 2 passed their turn.

Saboteur 2 passed their turn.

Plumber 1 moved to Pipe8

Plumber 1 moved to Cistern

Saboteur 1 passed their turn.

Plumber 2 passed their turn.

Saboteur 2 passed their turn.

Plumber 1 picked up New Pump 1 from the Cistern.

Plumber 1 moved to Pipe8

Saboteur 1 passed their turn.

Plumber 2 passed their turn.
```

Saboteur 2 passed their turn.

Plumber 1 moved to Pump4

Plumber 1 moved to Pipe6

Saboteur 1 passed their turn.

Plumber 2 passed their turn.

Saboteur 2 passed their turn.

Plumber 1 inserted a pump into Pipe6.

8.2.3 Saboteur and plumber changing water flow directions, plumber fixing a randomly broken down pump

- **Description**

A saboteur will change the input pipe to a pump to a pipe that has no water flowing through it, deliberately flawing the water flow of the system. The pump also randomly breaks down. Afterward, the plumber fixes the broken pump and re-changes the input pipe of that pump.

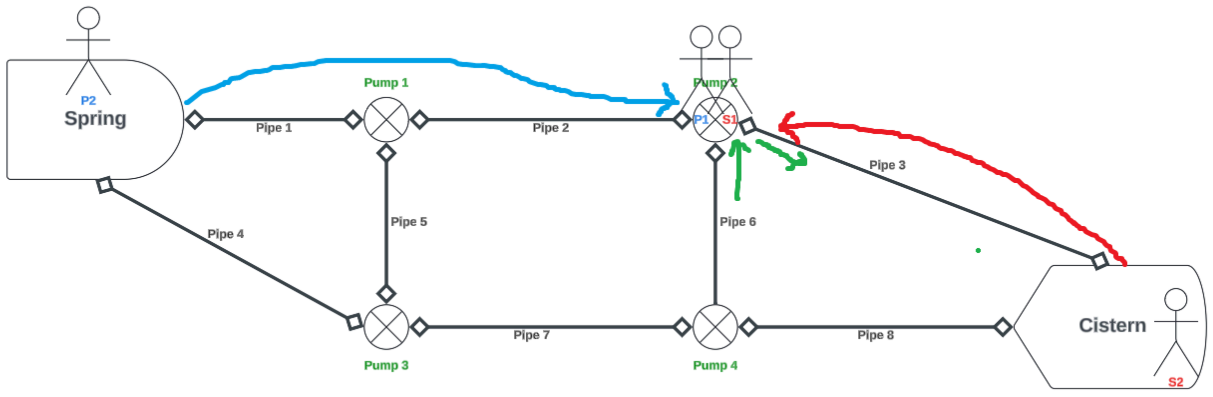


Fig 1. Plumber 1 and Saboteur 1 both move to Pump 2.
Saboteur 1 changes the input pipe of Pump 2 from Pipe 2 to Pipe 6. Since there is no water flowing through Pipe 6, Pipe 3 no longer receives water.

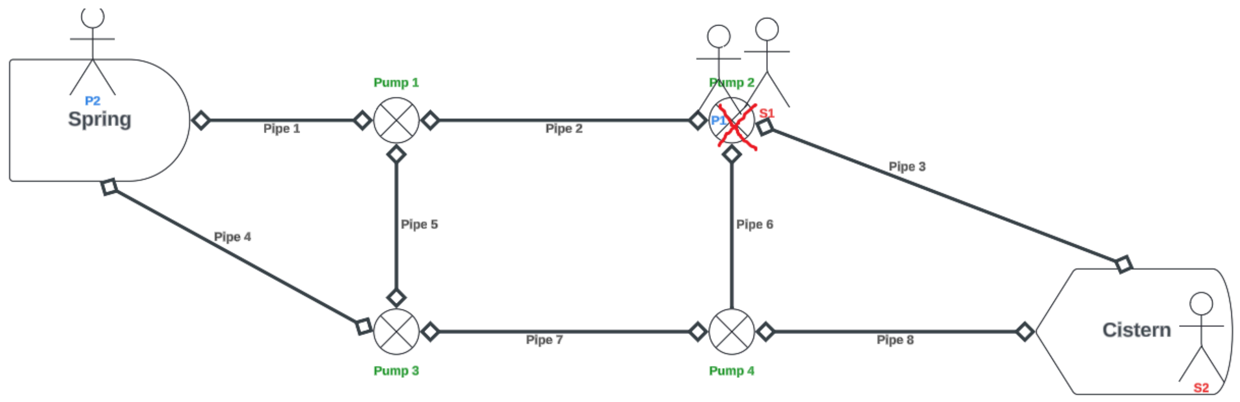


Fig 2. Additionally, the pump also breaks down randomly, meaning that it does not function properly anymore, even if the input pipe has water flowing in it.

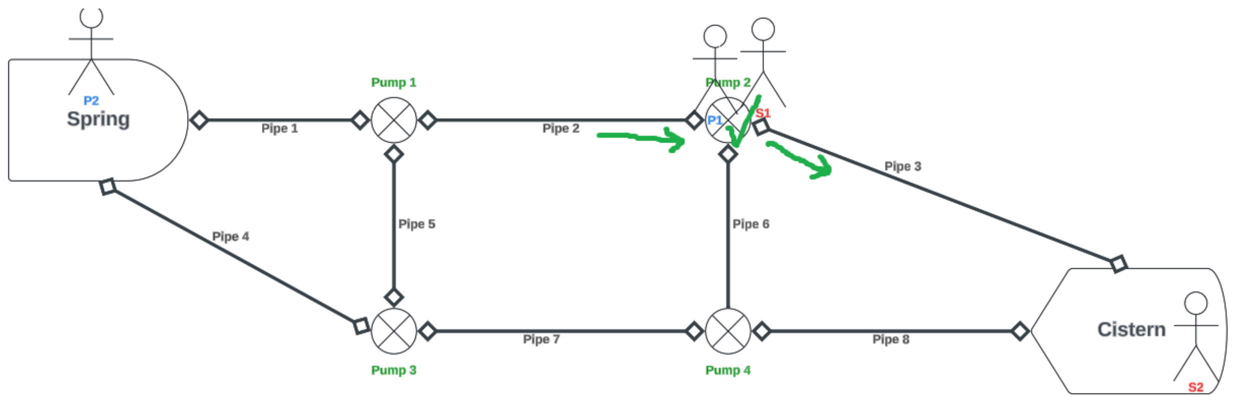


Fig 3. Plumber 1 fixes the broken pump and changes the input pipe of Pump 2 to Pipe 2 again, returning to the normal initial water flow.

- **Unit of functionality to be tested, possible failures**

The move(Element e) method which both players have.
The changeInputPipe(Pump p1, Pipe p2) method which both players have.
A pump randomly breaking down.
The fixPump(Pump pump) method of the plumber.

Possible failures include the pump not being fixed even after the player attempts to fix it, or the input pipe of that pump not changing.

- **Input**

4
Plumber 1
1
Plumber 2

1
Saboteur 1
Saboteur 2

Plumber 1: 1
Pipe1
1
Pump1

Saboteur 1: 1
Pipe3
1
Pump2

– *All other players pass their turns* –

Plumber 1: 1
Pipe2
1
Pump2

Saboteur 1: 2
Pipe6
5

– *Pump 2 now randomly breaks down* –
– *All other players pass their turns* –

Plumber 1: 4
8
Pipe2

- **Output**

Proceeding to the number of participating players menu.

4 players will participate in the game.

Proceeding to the username and team selection menu.

The name "Plumber 1" is validated.

Plumber 1 chose the Plumbers team.

The name "Plumber 2" is validated.

Plumber 2 chose the Plumbers team.

The name "Saboteur 1" is validated.

Saboteur 1 was automatically placed in the Saboteurs team.

The name "Saboteur 2" is validated.

Saboteur 2 was automatically placed in the Saboteurs team.

Initializing the game...

The game’s elements have been initialized successfully.

The game and timer have started!

Plumber 1 moved to Pipe1.

Plumber 1 moved to Pump1.

Saboteur 1 moved to Pipe3.

Saboteur 1 moved to Pump2.

Plumber 2 passed their turn.

Saboteur 2 passed their turn.

Plumber 1 moved to Pipe2.

Plumber 1 moved to Pump2.

Saboteur 1 changed the input pipe of Pump2 to Pipe6.

Saboteur 1 passed their turn.

Pump 2 randomly broke down.

Plumber 2 passed their turn.

Saboteur 2 passed their turn.

Plumber 1 fixed Pump2.

Plumber 1 changed the input pipe of Pump2 to Pipe2.

8.2.4 Initialization and starting the game

- **Description**

This test case will ensure that the processes of initializing and starting the game are done correctly. The game class has an initGame() method that sets up players and game elements, storing them in arrays. This method also calls the startGame() method which manages the turns for the players and starts the timer and actual gameplay.

- **Unit of functionality to be tested, possible failures**

The following methods are tested:
initGame(), startGame(), addPipe(), addPump(), addCistern(), addSpring()

Possible failures include faulty storage of elements in their respective arrays.

- **Input**

No explicit user input is required, all the players have to do is give the program player and team information, which is done as follows:

4

Plumber 1

1

Plumber 2

1

Saboteur 1

Saboteur 2

- **Output**

Output messages will be printed indicating that the initialization process of the players, teams, and the game itself was successfully completed.

Proceeding to the number of participating players menu.

4 players will participate in the game.

Proceeding to the username and team selection menu.

The name "Plumber 1" is validated.

Plumber 1 chose the Plumbers team.

The name "Plumber 2" is validated.

Plumber 2 chose the Plumbers team.

The name "Saboteur 1" is validated.

Saboteur 1 was automatically placed in the Saboteurs team.

The name "Saboteur 2" is validated.

Saboteur 2 was automatically placed in the Saboteurs team.

Initializing the game…

The game’s elements have been initialized successfully.

The game and timer have started!

8.2.5 Menu Input Validation Test

- **Description**

Testing the input validation of menus when user input is not of the format/type the program is expecting to process.

- **Unit of functionality to be tested, possible failures**

Interactions of the main menu, number of participating players menu, username choice menu, team selection menu, and Plumber and Saboteur actions menus.

Possible failures are a lack of proper input validation.

- **Input**

– Number of participating players menu –

Player: 5 (which is not allowed, there can only be 4 or 6 players playing).
Player: -4
Player: s
Player: 4. Four players play the game (2 vs. 2)

– Username and team selection menu –

Player 1: Joker
Player 1: 3
Player 1: 1. Plumber team

Player 2: LeeroyJenkins

Player 2: -5
Player 2: 2. Saboteur team

Player 3: Mike
Player 3. 1. Plumber team

Player 4: \n
Player 4: Oxlong

– Plumber actions menu –

Joker: 30
Joker: -8
Joker: 10. Pass Turn

– Saboteur actions menu –

LeeroyJenkins: 7
LeeroyJenkins: -3
LeeroyJenkins: 5. Pass Turn

- **Output**

Proceeding to the number of participating players menu.

Invalid input, please choose one of the valid options (4 or 6).

Invalid input, please choose one of the valid options (4 or 6).

Invalid input, please choose one of the valid options (4 or 6).

4 players will participate in the game.

Proceeding to the username and team selection menu.

The name "Joker" is validated.

Invalid input, please choose one of the valid options (1 or 2).

Joker chose the Plumbers team.

The name "LeeroyJenkins" is validated.

Invalid input, please choose one of the valid options (1 or 2).

LeeroyJenkins chose the Saboteurs team.

The name "Mike" is validated.

```
Mike chose the Plumbers team.

Invalid input, please enter a valid name.

The name "Oxlong" is validated.

Oxlong was automatically placed in the Saboteurs team.

Initializing the game...

The game’s elements have been initialized successfully.

The game and timer have started!

Invalid input, please choose one of the valid options (1-11).

Invalid input, please choose one of the valid options (1-11).

Player Joker passed their turn.

Invalid input, please choose one of the valid options (1-6).

Invalid input, please choose one of the valid options (1-6).

Player LeeroyJenkins passed their turn.
```

8.2.6 Multiple players testing prevented actions such as fixing an already working pump, as well as normal actions like puncturing and fixing pipes

- **Description**

In this test case, players alternate turns, performing actions such as moving, puncturing pipes, and fixing pipes. Saboteur 1 moves to sabotage pipes, while Plumbers 1 and 2 work to repair them. This test handles scenarios like attempting to fix an already working pump and moving to non-standable elements.

Fig 1:
Player P1 moves to pump 3
Player S1 moves to pump 2

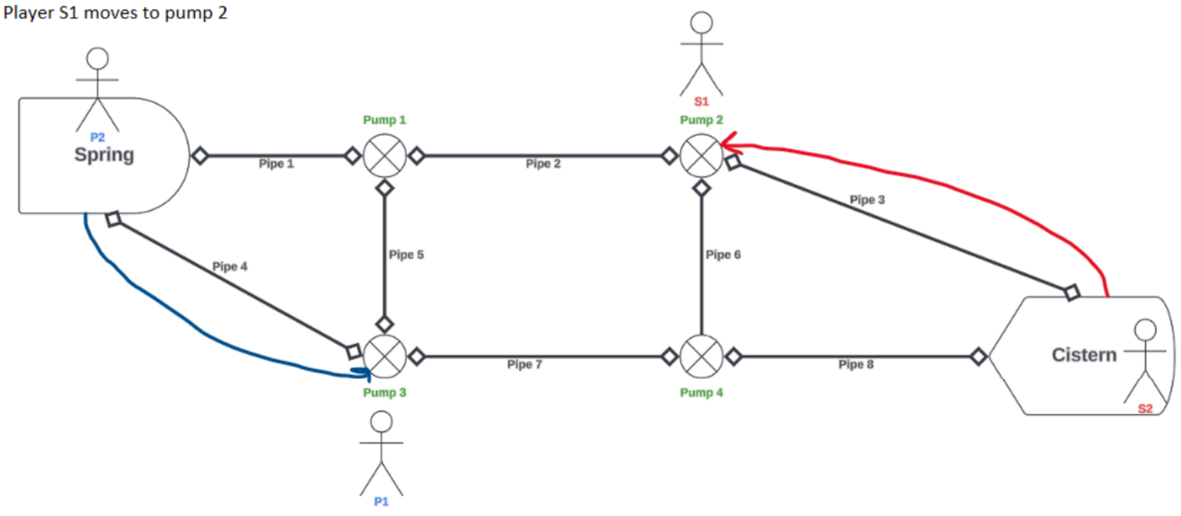


Fig 2:
P2 moves to pump 1
S1 moves to pipe 2 and punctures it

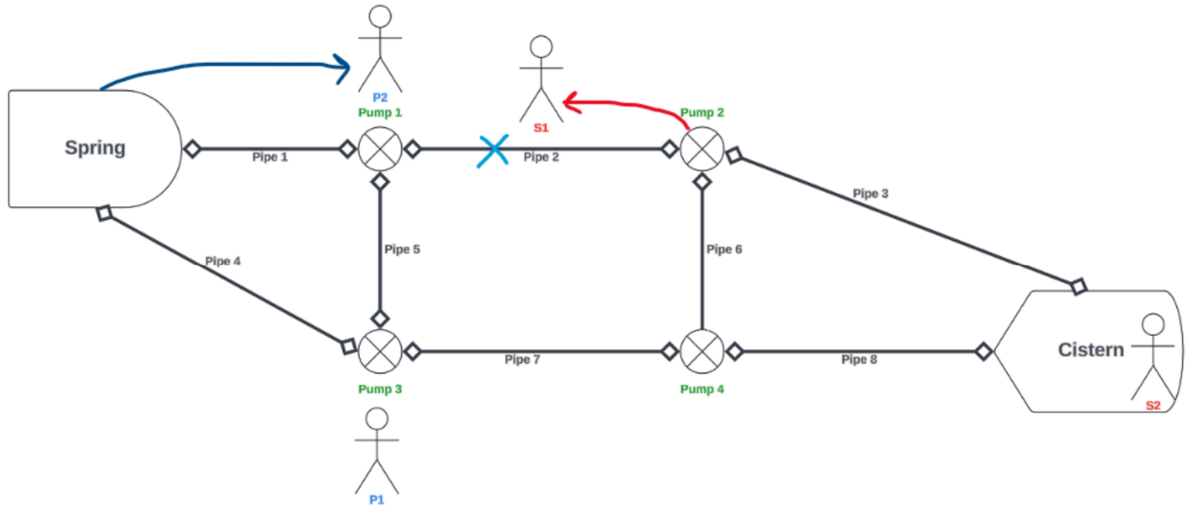


Fig 3:
S1 moves to pipe 5 and punctures it
P2 moves to pipe 2 and fixes it

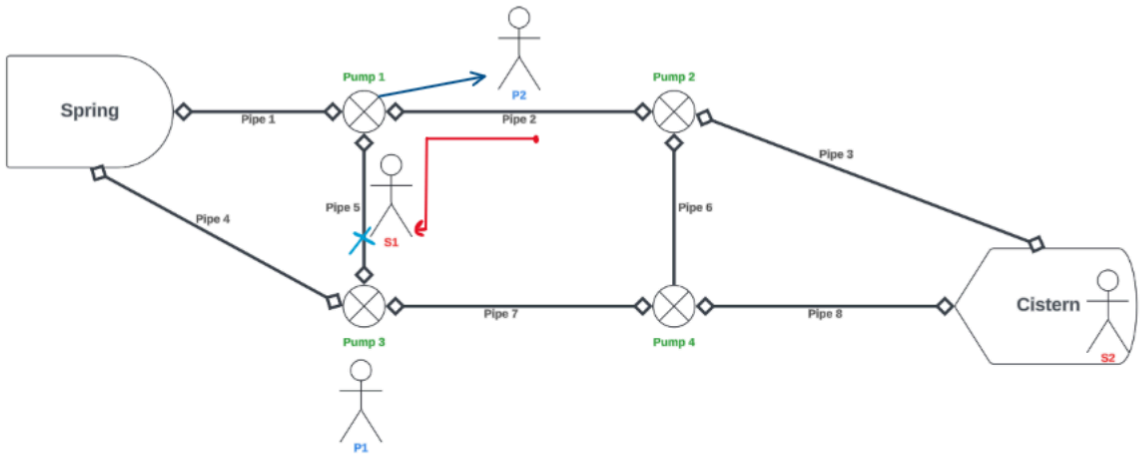
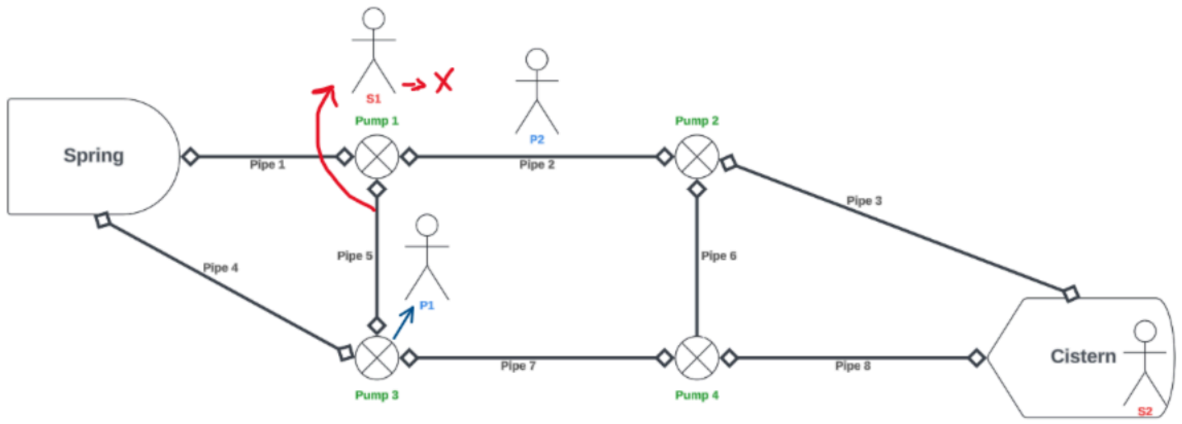


Fig 4:
S1 moves to pump 1 and tries to advance to pipe 2, but its occupied
P1 takes turn and moves to pipe 5 to fix it



- **Unit of functionality to be tested, possible failures**

The move(Element e) method which both players have.
The feature that prevents players from moving to pipes if they are occupied.
The feature that prevents players from fixing an element that is already in working condition.
The puncture(Pipe p1) method which the saboteur has.
The fixPipe(Pipe p) method of the Plumber.
The fixPump(Pump pump) method of the Plumber.

Possible failures include the pipe not being punctured/fixed even after the players issue the commands, players moving to non-standable or occupied elements, or being able to fix already working elements such as a working pump.

- **Input**

4
Plumber 1
1
Plumber 2
1
Saboteur 1
Saboteur 2
Plumber 1: 1
Pipe4
1
Pump3

Saboteur 1:
1
Pipe3
1
Pump2

Plumber 2:
1
Pipe1
1
Pump1

Saboteur 2:
5

Plumber 1:
10

Saboteur 1:
1
Pipe2
4

Plumber 2:
10

Saboteur 2:
5

Plumber 1:
10

Saboteur 1:
1
Pump1
1
Pipe5

Plumber 2:
4 “It's already working so it should prevent the player from fixing it”
1
Pipe2
Saboteur 2:
5

Plumber 1:
10

Saboteur 1:
5

Plumber 2:
5
10

Saboteur 2:
5

Plumber 1:
10

Saboteur 1:
4
1
Pump1

Plumber 2:
10

Saboteur 2:
5

Plumber 1:
10

Saboteur 1:
1
Pipe2 “Can't move to it (occupied)”
5

Plumber 2:
10

Saboteur 2:
5

Plumber 1:
1
Pipe5
5

- **Output**

```
Proceeding to the number of participating players menu.
4 players will participate in the game.
Proceeding to the username and team selection menu.
The name "Plumber 1" is validated.
Plumber 1 chose the Plumbers team.
The name "Plumber 2" is validated.
Plumber 2 chose the Plumbers team.
The name "Saboteur 1" is validated.
Saboteur 1 was automatically placed in the Saboteurs team.
The name "Saboteur 2" is validated.
Saboteur 2 was automatically placed in the Saboteurs team.
Initializing the game...
The game’s elements have been initialized successfully.
The game and timer have started!
Plumber 1 moved to Pipe4
Plumber 1 moved to Pump3
```

```
Saboteur 1 moved to Pipe3
Saboteur 1 moved to Pump2
Plumber 2 moved to Pipe1
Plumber 2 moved to Pump1
Saboteur 2 passed their turn.
Plumber 1 passed their turn.
Saboteur 1 moved to Pipe2
Saboteur 1 punctured Pipe2
Plumber 2 passed their turn.
Saboteur 2 passed their turn.
Plumber 1 passed their turn.
Saboteur 1 moved to Pump1
Saboteur 1 moved to Pipe5
Plumber 2 attempted to fix Pump1, but it's already working.
Plumber 2 moved to Pipe2
Saboteur 2 passed their turn.
Plumber 1 passed their turn.
Saboteur 1 passed their turn.
Plumber 2 fixed Pipe2
Plumber 2 passed their turn.
Saboteur 2 passed their turn.
Plumber 1 passed their turn.
Saboteur 1 punctured Pipe5
Saboteur 1 moved to Pump1
Plumber 2 passed their turn.
Saboteur 2 passed their turn.
Plumber 1 passed their turn.
This location is currently occupied. Choose another location.
Saboteur 1 passed their turn.
Plumber 2 passed their turn.
Saboteur 2 passed their turn.
Plumber 1 moved to Pipe5
Plumber 1 fixed Pipe5
```

8.2.7 New pipe randomly being manufactured at the cistern, picking up an end of pipe and inserting it into another pipe to redirect water

- **Description**

A new pipe is being manufactured. One end of the pipe is connected to the cistern, while the other end remains free. This means that water cannot flow through the pipe unless it is connected to a pump via the end of the pipe. The plumber takes the free end of an existing pipe, inserts it into the new one, and redirects the pump's output to the newly manufactured pipe.

Fig 1:
New pipe manufactured at the cistern connected to pump 2 with a free end

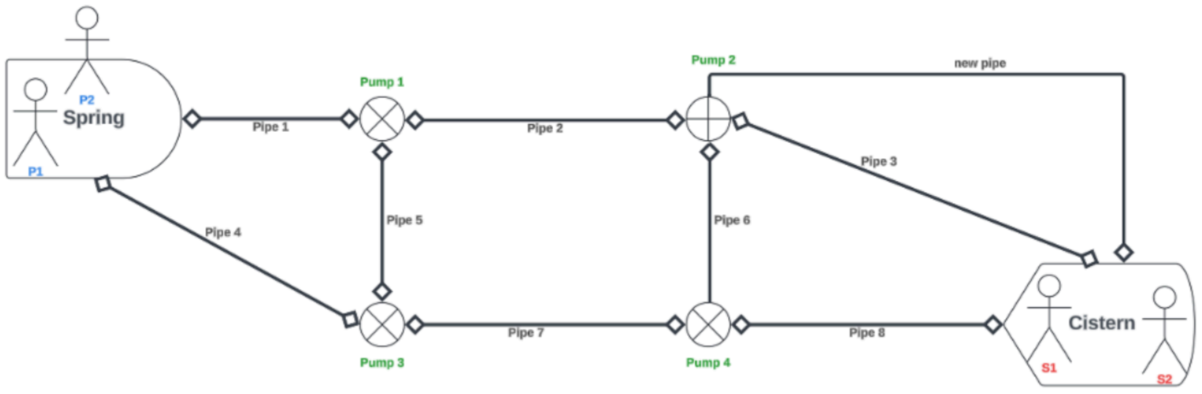


Fig 2 :
P1 moves to pipe 6 to get the end of pipe

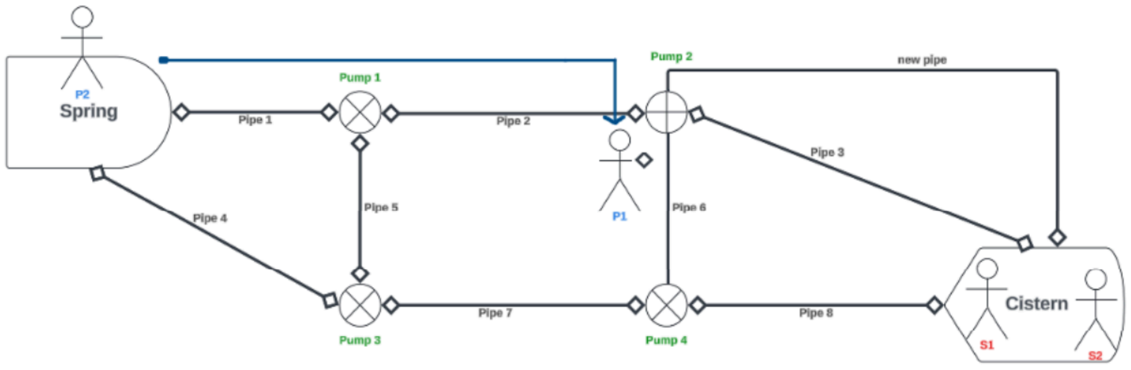
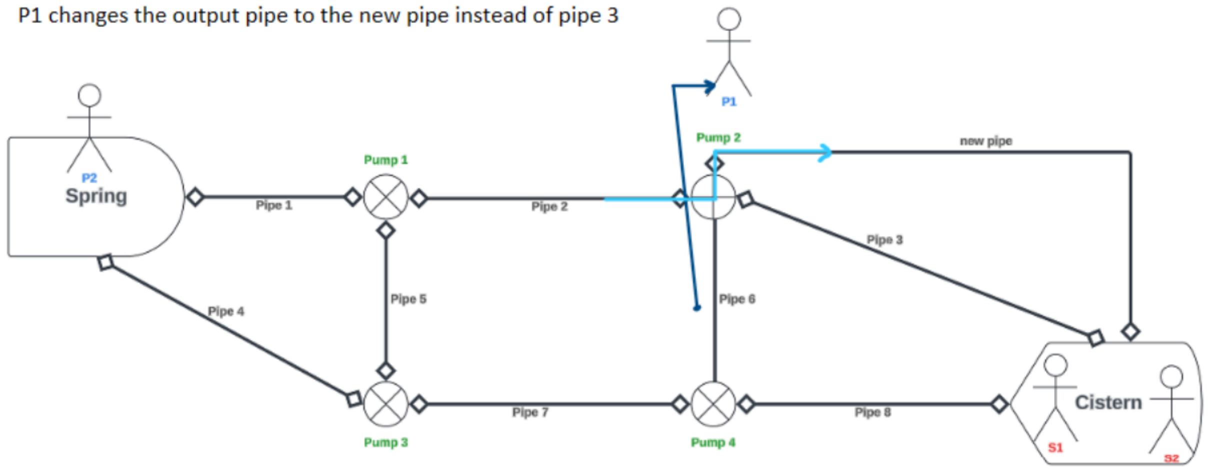


Fig 3:
P1 moves the new pipe and connect the end of pipe to it
P1 changes the output pipe to the new pipe instead of pipe 3



- **Unit of functionality to be tested, possible failures**

ManufacturePipe() method which generates a new pipe at the cistern
The move(Element e) method which both players have.
The getEnd(EndOfPipe EoP) method which allows a plumber to pick up an end of pipe object from the end of a pipe
The insertPipeEnd(Element e) method which allows the plumber to connect an end of pipe object to an element in the game.
The changeOutputPipe(Pump p1, Pipe p2) method.

Possible failures:
The new pipe is not generated correctly at the cistern.
The Plumber is unable to pick up the end of a pipe.
Inability to connect the end of a pipe to another element.
The pump's output is not successfully redirected to the new pipe.

- **Input**

4
Plumber 1
1
Plumber 2
1
Saboteur 1
Saboteur 2

Plumber 1:
1
Pipe1
1
Pump1

Saboteur 1:
5

Plumber 2:
10

Saboteur 2:
5

Plumber 1:
1
Pipe2
1
Pump2

Saboteur 1:
5

Plumber 2:
10

Saboteur 2:
5

Plumber 1:
6
Pipe6
7
New Pipe 1

Saboteur 1:
5

Plumber 2:
10

Saboteur 2:
5

Plumber 1:
9
New Pipe 1

- **Output**

Proceeding to the number of participating players menu.

4 players will participate in the game.

Proceeding to the username and team selection menu.

The name "Plumber 1" is validated.

Plumber 1 chose the Plumbers team.

The name "Plumber 2" is validated.

Plumber 2 chose the Plumbers team.

The name "Saboteur 1" is validated.

Saboteur 1 was automatically placed in the Saboteurs team.

The name "Saboteur 2" is validated.

Saboteur 2 was automatically placed in the Saboteurs team.

Initializing the game...

The game's elements have been initialized successfully.

The game and timer have started!

Plumber 1 moved to Pipe1

Plumber 1 moved to Pump1

Saboteur 1 passed their turn.

Plumber 2 passed their turn.

Saboteur 2 passed their turn.

```
Plumber 1 moved to Pipe2
Plumber 1 moved to Pump2
Saboteur 1 passed their turn.
Plumber 2 passed their turn.
Saboteur 2 passed their turn.
Plumber 1 picked up the end of Pipe6 connected to Pump2
Plumber 1 inserted the end of pipe into New Pipe 1
Saboteur 1 passed their turn.
Plumber 2 passed their turn.
Saboteur 2 passed their turn.
Plumber 1 changed the output pipe of Pump2 to New Pipe 1
```

8.2.8 End Game and Winning Conditions Evaluation

Description: This test case evaluates the functionality of the "End Game" option. Players perform dummy actions, including movement and turn passing. At a certain point, a player chooses to end the game, triggering the automatic invocation of the "EndGame()" method. The expected outcome, in this specific test case, is for Team Plumbers to win. Additionally, regardless of the expected outcome, checks for determining the winning team will be performed to ensure the accuracy of the game's conclusion. These checks include comparing the total collected water against the total leaked water, with messages displayed based on the comparison.

Please note that the game can also end if the Game Timer expires, even if none of the players choose the "End the game" option.

Unit of Functionality to be Tested:
EndGame() method: Triggers the end of the game when a player chooses the option.
Move(Element e) and takeTurn(): Allows players to move and pass turns as dummy actions.
determineWinner()
calculateLeakedWater()
calculateCollectedWater()

Possible Failures:

EndGame() method failure: The game does not correctly end when the "End Game" option is chosen.

Movement and Turn Passing failure: Players are unable to perform basic movement and turn-passing.
Incorrect computation of leaked or collected water, leading to inaccurate determination of the winning team.

- **Input**

4
Plumber 1
1
Plumber 2
1
Saboteur 1
Saboteur 2

Plumber 1:
1 "Move to an element"
Pipe1
1
Pump1

Saboteur 1:
1
Pipe3
1
Pump2

Plumber 2:
1
Pipe1
1
Pump1

Saboteur 2: 5

Plumber 1: 11

- **Output**

Proceeding to the number of participating players menu.

4 players will participate in the game.

```
Proceeding to the username and team selection menu.
The name "Plumber 1" is validated.
Plumber 1 chose the Plumbers team.
The name "Plumber 2" is validated.
Plumber 2 chose the Plumbers team.
The name "Saboteur 1" is validated.
Saboteur 1 was automatically placed in the Saboteurs team.
The name "Saboteur 2" is validated.
Saboteur 2 was automatically placed in the Saboteurs team.
Initializing the game...
The game’s elements have been initialized successfully.
The game and timer have started!
Plumber 1 moved to Pipe1
Plumber 1 moved to Pump1
Saboteur 1 moved to Pipe3
Saboteur 1 moved to Pump2
Plumber 2 moved to Pipe1
Plumber 2 moved to Pump1
Saboteur 2 passed their turn.
The timer has stopped, and the game has ended. The results are being evaluated...
Total water leaked was 0, and total water collected was 16. The Plumbers won the game.
```

Note: *“ Upon execution of the "EndGame()" method, the output will display a message indicating the end of the game and the winning team based on the comparison of collected and leaked water values. The output message will be determined by the following logic:*

If the total collected water exceeds the total leaked water, the message will be:
"Game over! Team Plumbers win!"
If the total leaked water exceeds the total collected water, the message will be:
"Game over! Team Saboteurs win!"
If the total collected water equals the total leaked water, indicating a draw, the message will be:
***"Game over! It's a draw!"** ”*

“The expected outcome is for the Plumbers to win the game since there is an initial water flow and no leakage when the game is ended using the "End Game" option. “