A-Maze-ingly Retro Route Puzzle

Requisites:

Docker is a mandatory technology to master. Each artifact must

1. contains a Dockerfile into the root directory
2. the full directory will be mounted under `/mnt/` folder into the docker image builded from your docker file
3. if the implementation is for a network accessible service it must be binded to the port :`9090`
4. contains a **build** script runnable within the docker container generated from the Docker file named `scripts/build.sh`
5. contains a **test** script runnable within the docker container generated from the Docker file named `scripts/test.sh`
6. contains a **run** script runnable within the docker container generated from the Docker file named `scripts/run.sh`

The candidate can simulate the review process with these commands, that must be run from the root of the project folder:

commands

```
docker build -t mytest .
docker run -v $(pwd):/mnt -p 9090:9090 -w /mnt mytest ./scripts/build.sh
docker run -v $(pwd):/mmt -p 9090:9090 -w /mnt mytest ./scripts/tests.sh
docker run -v $(pwd):/mnt -p 9090:9090 -w /mnt mytest ./scripts/run.sh
```

Problem:

Write a program that will output a valid route one could follow to collect all specified items within a map. The map is a json description of set of rooms with allowed path and contained object.

exercize starts with an input of:

- json reppresentation of map
- starting room
- list of object to collect

```
Room type allowed fields
  id: Integer
  name: String
  north: Integer //referring to a connected room
  south: Integer //referring to a connected room
  west: Integer  //referring to a connected room
  east: Integer  //referring to a connected room
  objects: List  //of Objects

Object type allowed fields
  name: String //Object Name
```

Example 1:

Map

```
{
        "rooms": [
                { "id": 1, "name": "Hallway", "north": 2, "objects": [] },
                { "id": 2, "name": "Dining Room", "south": 1, "west": 3,
"east": 4, "objects": [] },
                { "id": 3, "name": "Kitchen","east":2, "objects": [ {
"name": "Knife" } ] },
                { "id": 4, "name": "Sun Room","west":2, "objects": [ {
"name": "Potted Plant" } ] }
        ]
}
```

Input Start Room ID= 2

Input Objects To Collect= Knife, Potted Plant

Output

```
ID  Room          Object collected
---------------------------------
2   Dining Room   None
1   Hallway       None
2   Dining Room   None
3   Kitchen       Knife
2   Dining Room   None
4   Sun Room      Potted Plant
```

Example 2

Map

```
{
        "rooms": [
            { "id": 1, "name": "Hallway", "north": 2, "east":7, "objects":
[] },
            { "id": 2, "name": "Dining Room", "north": 5, "south": 1,
"west": 3, "east": 4, "objects": [] },
            { "id": 3, "name": "Kitchen","east":2, "objects": [ { "name":
"Knife" } ] },
            { "id": 4, "name": "Sun Room","west":2, "north":6, "south":7,
"objects": [] },
            { "id": 5, "name": "Bedroom","south":2, "east":6, "objects":
[{ "name": "Pillow" }] },
            { "id": 6, "name": "Bathroom","west":5, "south":4, "objects":
[] },
                { "id": 7, "name": "Living room","west":1, "north":4,
"objects": [{ "name": "Potted Plant" }] }
        ]
}
```

Input Start Room ID = 4

Input Objects To Collect= Knife, Potted Plant, Pillow

Output

```
ID  Room          Object collected
---------------------------------
4   Sun Room      None
6   Bathroom      None
4   Sun Room      None
7   Living room   Potted Plant
4   Sun Room      None
2   Dining Room   None
5   Bedroom       Pillow
2   Dining Room   None
1   Hallway       None
2   Dining Room   None
3   Kitchen       Knife
```

Goals:

- TDD approach.
- Build a Docker container with runnable code inside so that we can mount a volume in it and test on different maps.