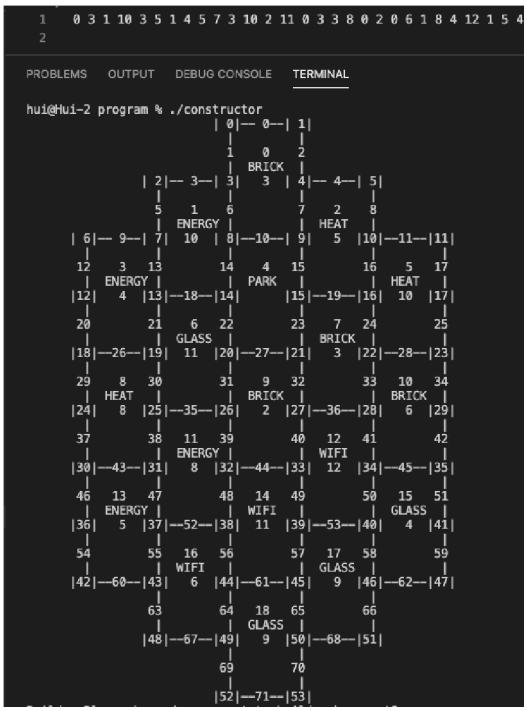
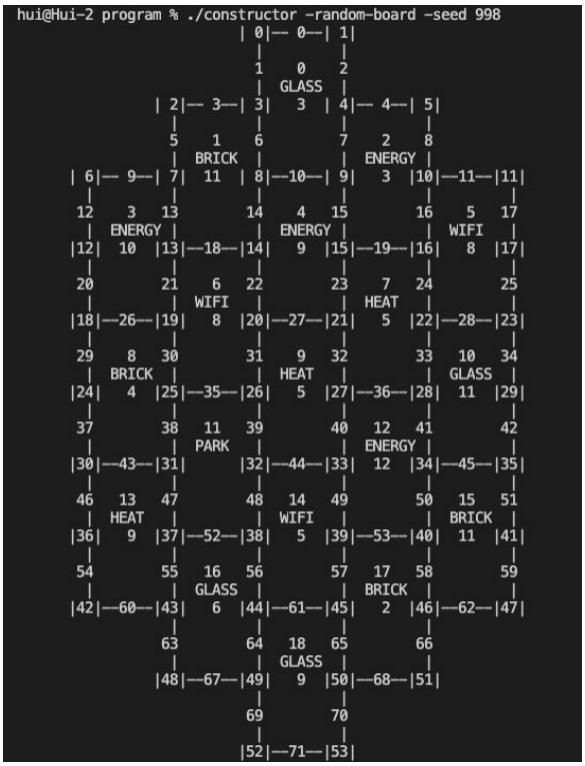
Demo

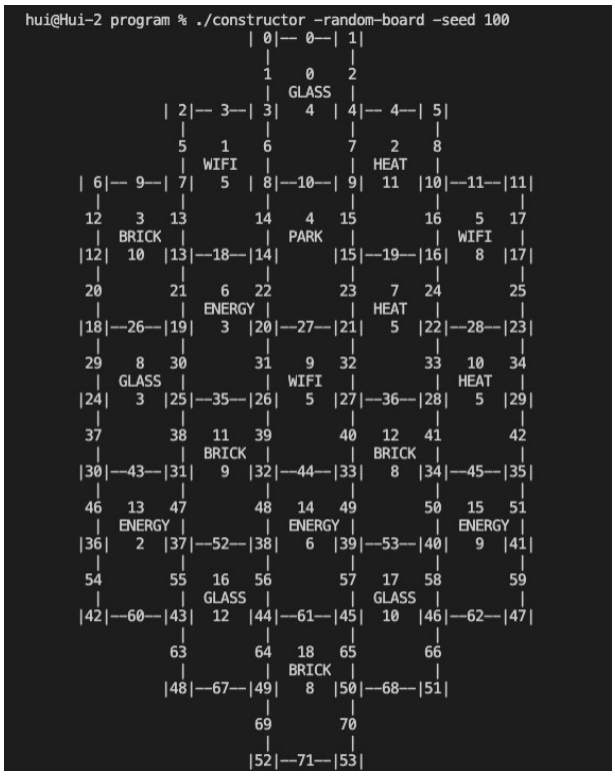
Set up of the Game



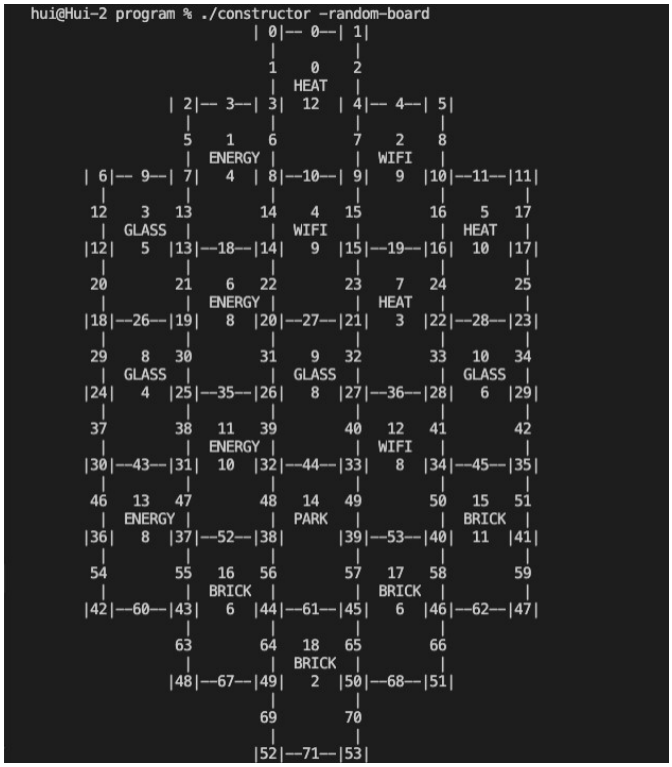
the following command “-load xxx”, “-board xxx” and “random-board”, where “xxx” represents the filename. If you run this game without inputting anything, the game will use the default board layout (layout.txt shown at the top of the picture to the left)

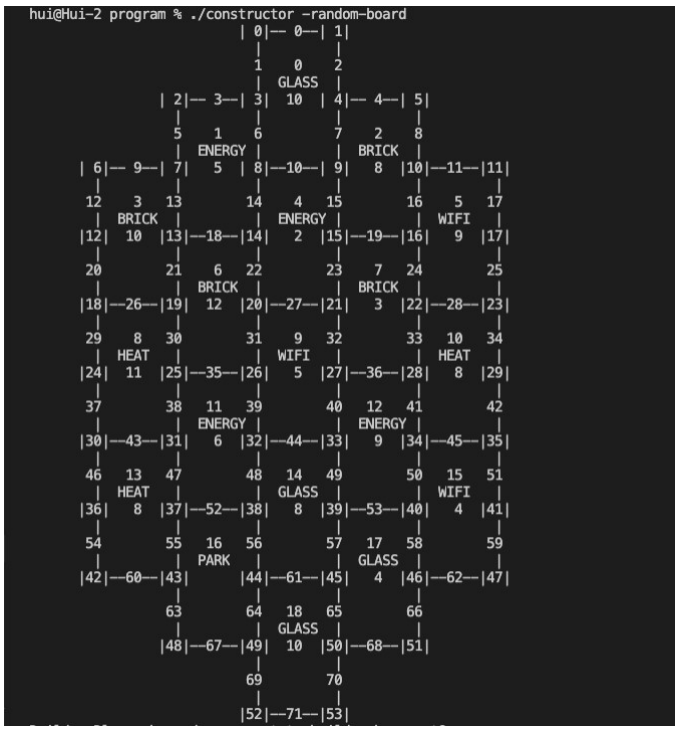
When using the command line argument “-random-board”, “-seed xxx” is optional. If the seed is the same, then you will always get the same board. If you change the seed, you will get a different board as the following two pictures shown.

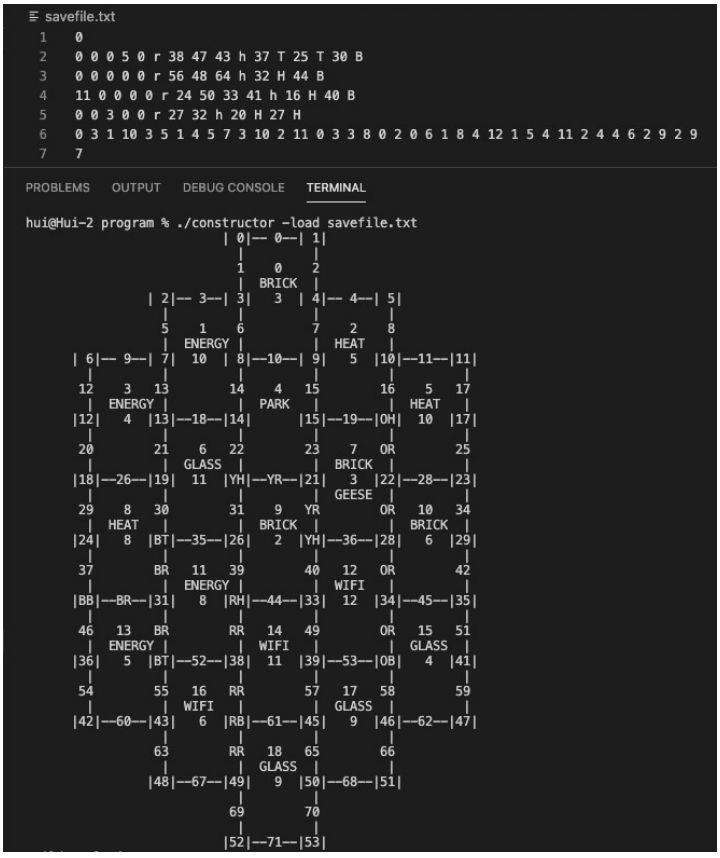




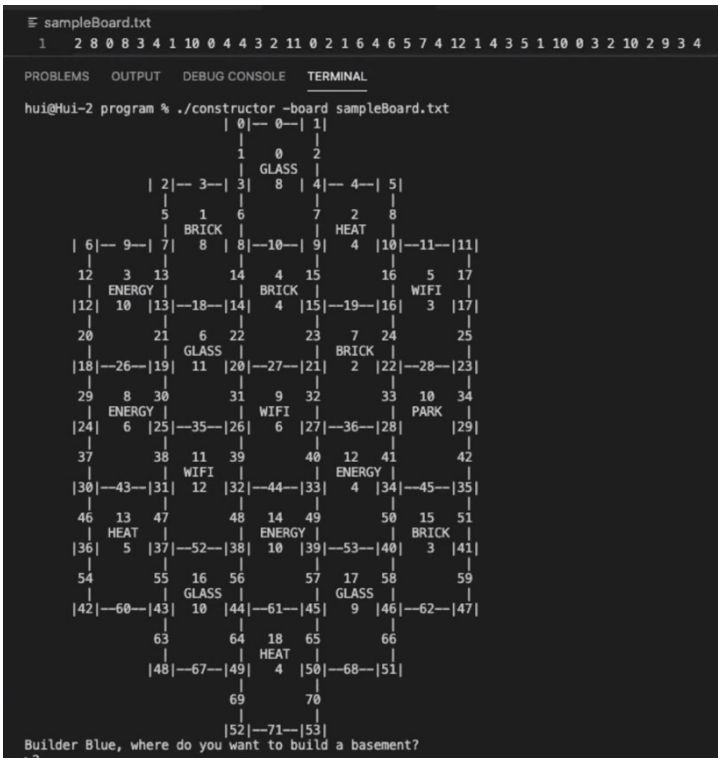
However, if you dose not input “-seed xxx”, we use the current time as the seed, which indicates that you will not get the same board every time you play the game.







"-load xxx": represents that the game will be loaded from a saved file xxx. (savefile.txt shown at the top of the picture to the left)



"-board xxx" represents loading the game with the board specified in the file xxx. (sampleBoard.txt shown at the top of the picture shown to the left)

We will use the default board layout to display all commands in the game.

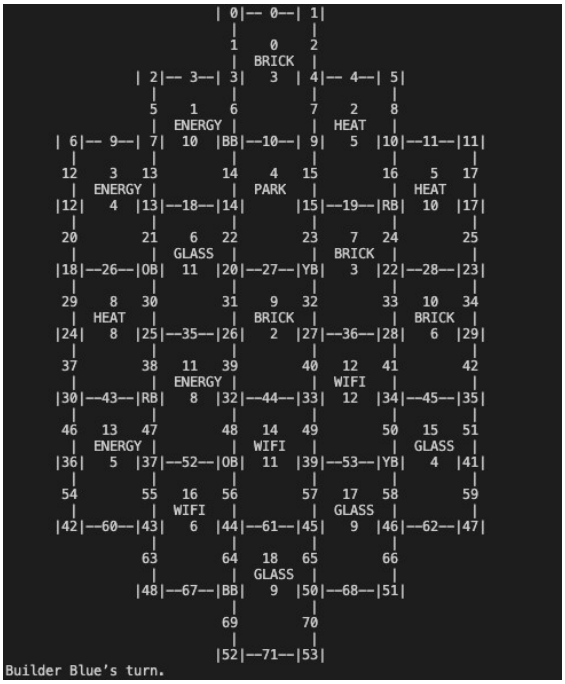
```
Builder Blue, where do you want to build a basement?
>8
Builder Red, where do you want to build a basement?
>109
You cannot build here.
location does not exist.
Builder Red where do you want to build a basement?
>cs
ERROR: not an integer
Builder Red where do you want to build a basement?
>9
You cannot build here.
Basements already exist as locations: 8
Builder Red where do you want to build a basement?
```

At the beginning of the game, players are required to input the location of their house. If the input is not valid (i.e. not an integer or a number out of scope), they are required to choose the location until it is valid.

```
Builder Blue, where do you want to build a basement?
>8
Builder Red, where do you want to build a basement?
>16
Builder Orange, where do you want to build a basement?
>19
Builder Yellow, where do you want to build a basement?
>21
Builder Yellow, where do you want to build a basement?
>40
Builder Orange, where do you want to build a basement?
>38
Builder Red, where do you want to build a basement?
>31
Builder Blue, where do you want to build a basement?
>49
```

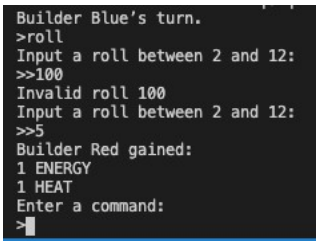
Also, at the start of the game, players should follow the sequence "Blue-Red-Orange-Yellow-Yellow-Orange - RedBlue" to choose their first two residences.

Beginning of the turn:

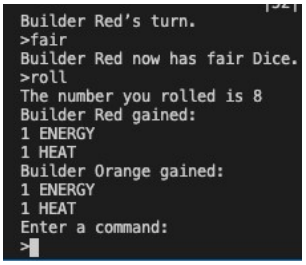


After all players have successfully build their first two residences, the current board will be displayed immediately.

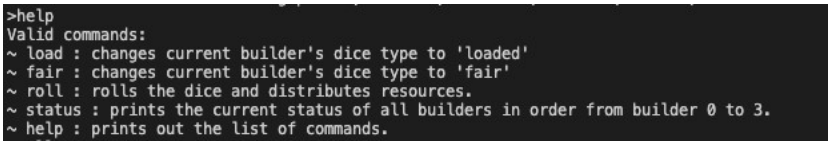
Each player can choose whether to use a fair dice or a loaded dice at the beginning of each turn. Once they have already chosen the type of dice, the setting of the dice will be saved until the player changes it for the next time.



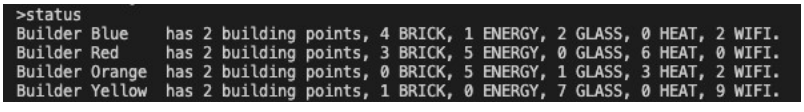
Since they must choose their first two residences, default type of is loaded dice. If the player chooses to use a loaded dice, he/she must input a number between 2 and 12 inclusively. Otherwise, the player will be prompted to input until there is a valid one.



If the player chooses to use a fair dice, the game will randomly generate the points. After rolling the dice, players will receive a message of "<which player> gained <how many resources>".



In addition, if players are confused about what commands they can use, inputting "help" allows them to see all valid commands. ("load", "fair", "roll", "status" and "help")



"status" displays all information of each player regarding the total points, resources they gained.

During the turn

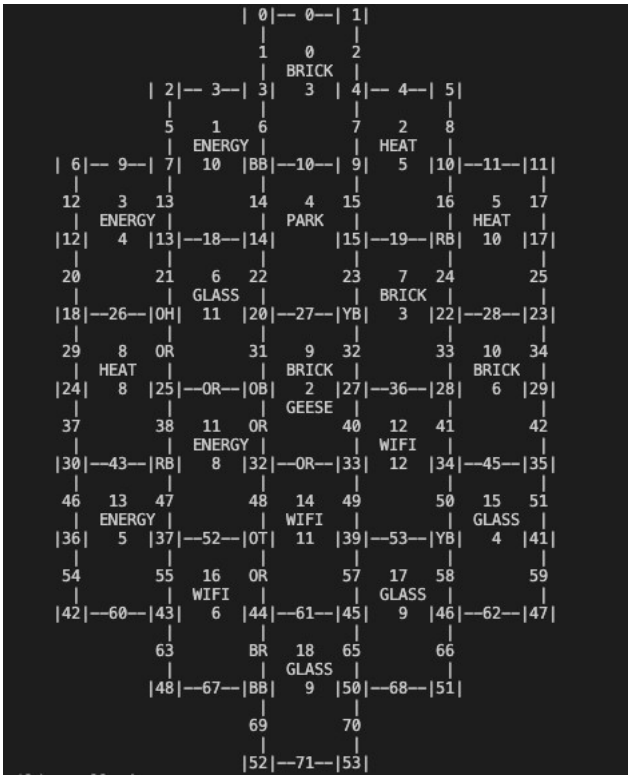
Players can input any of the following commands:

"board", "status", "residences", "build-road <road#>", "build-res <housing#>", "improve <housing#>", "trade <colour> <give> <take>", "next", "save <file>" and "help".

If players are confused about what commands they can use, inputting “help” allows them to see all valid commands.

```
>help
Valid commands:
~ board : prints the current board.
~ status : prints the current status of all builders in order from builder 0 to 3.
~ residences : prints the residences the current builder has currently completed.
~ build - road <road#> : attempts to builds the road at <road#>.
~ build - res <housing#> : attempts to builds a basement at <housing#>.
~ improve <housing#> : attempts to improve the residence at <housing#>.
~ trade <colour> <give> <take> : attempts to trade with builder <colour>, giving one resource of type <give> and receiving one resource of type <take>.
~ next : passes control onto the next builder in the game.
~ save <file> : saves the current game state to <file>.
~ help : prints out the list of commands.
```

“board” allows the player to see the current situation of the game (functioned as a map)



In order to describe remaining methods, we name this picture as graph 1. “status” remains the same functionality as described above.

```
>status
Builder Blue    has 2 building points, 4 BRICK, 1 ENERGY, 2 GLASS, 0 HEAT, 2 WIFI.
Builder Red     has 2 building points, 3 BRICK, 5 ENERGY, 0 GLASS, 6 HEAT, 0 WIFI.
Builder Orange  has 2 building points, 0 BRICK, 5 ENERGY, 1 GLASS, 3 HEAT, 2 WIFI.
Builder Yellow  has 2 building points, 1 BRICK, 0 ENERGY, 7 GLASS, 0 HEAT, 9 WIFI.
```

“residences” allows the active player to see the information of all residences he/she has built. (including the location and the type of each residence)

```
>residences
Orange has built:
19 H
26 B
38 T
Enter a command:
```

“build-road <road#>”, where <road#> stands for the position of the road, allows players to build a road at the position <road#>.

Based on the rule of this game, if a player tries to build a road at xx, there are three possible outcomes.

- Successfully built a road at the position.
- You do not have enough resources.
- You cannot build here.

```
>build-road 39
>Builder Orange successfully built a Road at 39
Enter a command:
>build-road 44
>Builder Orange successfully built a Road at 44
Enter a command:
>build-road 48
>You do not have enough resources.

The cost of a Road is one HEAT and one WIFI resource.
Enter a command:
>build-road 55
>You cannot build here.
Enter a command:
```

Similarly, “build-res <housing#>”, where <housing#> stands for the position of the residence, allows players to build a house at <housing#>.

Based on the rule of this game, if a player tries to build a residence at <housing#>, there are three possible outcomes.

- You do not have enough resources.

```
>build-res 33
>
You do not have enough resources.

The cost of a Basement is one BRICK, one ENERGY, one GLASS, and one WIFI resource.
```

- You cannot build here.
  Player cannot build at 39 since Yellow has a basement at 40 (a neighbor).
  Player cannot build at 40 since Yellow has a basement at 40 (itself).
  Player cannot build at 15 since there is no neighbor road build by the same player.
- Successfully built a basement at xx.

```
>build-res 39
>You cannot build here.
Enter a command:
>build-res 40
>You cannot build here.
Enter a command:
>build-res 33
>Builder Yellow successfully built a Basement at 33.
Enter a command:
>build-res 15
>You cannot build here.
Enter a command:
```

After building a road at 39, 44 and building a residence at 33, the board becomes graph1 shown in the previous page.

“improve <housing#>”, where <housing#> stands for the position of the residence, allows players to improve the residence located at <housing#>.

Based on the rule of this game, if a player tries to improve the residence at <housing#>, there are three possible outcomes.

- Successfully built a <type of the residence after improving it> at the position
  At 21, House -> Tower
  At 40, Basement -> House -> Tower
- You cannot improve here. location does not exist.
  59 out of scope
- You can’t improve that building.
  Residence at 33 is already a Tower
- You do not have enough resources.

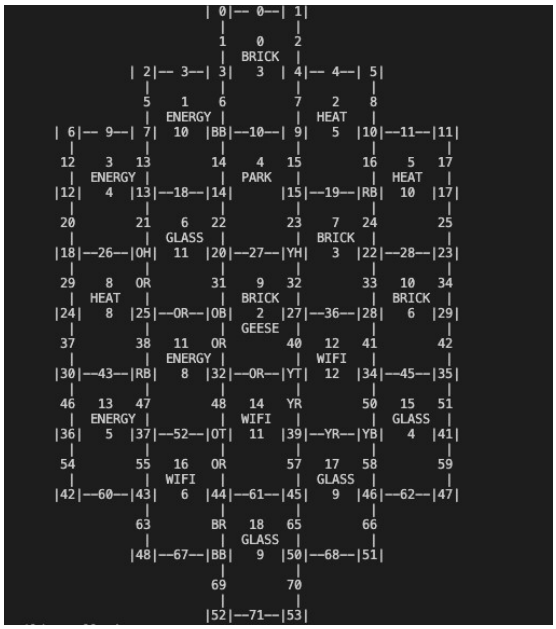
```
>residences
Yellow has built:
21 H
33 T
40 B
Enter a command:
>improve 21
>Builder Yellow successfully built a Tower at 21.
Enter a command:
>improve 33
>You can't improve that building.
Enter a command:
>improve 40
>Builder Yellow successfully built a House at 40.
Enter a command:
>improve 40
>Builder Yellow successfully built a Tower at 40.
Enter a command:
>improve 59
>ERROR:  You cannot improve here. location does not exist.
>status
ERROR:  isn't a valid integer.
>50
Invalid residence.
```

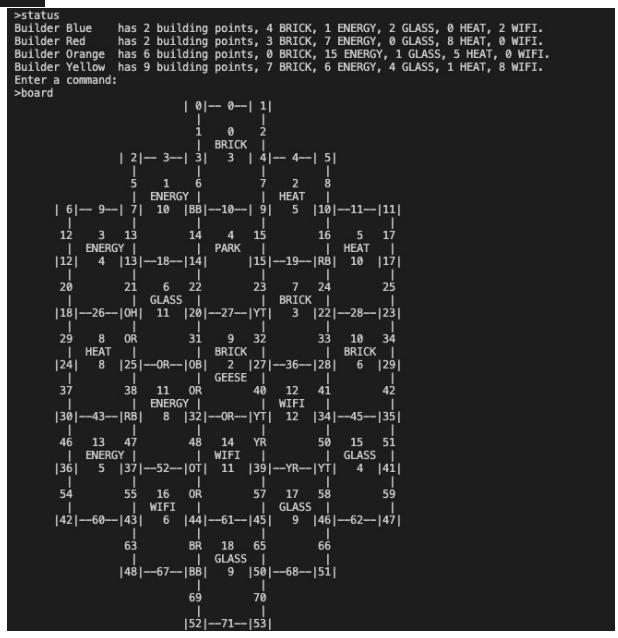
```
>improve 40
>You do not have enough resources.

The cost to improve a Basement to a House is two GLASS and three HEAT resource.
The cost to improve a House to a Tower is three BRICK, two ENERGY, two GLASS, one WIFI, and two HEAT.
```

Before improving the residences (left) and after improving the residences (right)

```
>status
Builder Blue    has 2 building points, 4 BRICK, 1 ENERGY, 2 GLASS, 0 HEAT, 2 WIFI.
Builder Red     has 2 building points, 3 BRICK, 7 ENERGY, 0 GLASS, 8 HEAT, 0 WIFI.
Builder Orange  has 6 building points, 0 BRICK, 15 ENERGY, 1 GLASS, 5 HEAT, 0 WIFI.
Builder Yellow  has 6 building points, 10 BRICK, 8 ENERGY, 8 GLASS, 6 HEAT, 9 WIFI.
```

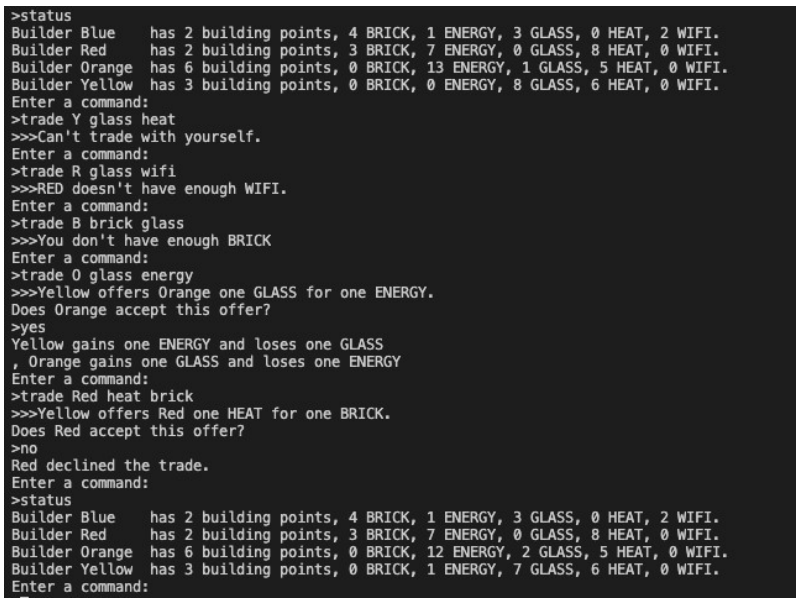




“trade(<colour> <give> <take>)” allows the current player to trade with <colour> giving one resource of type <give> and receiving one resource of type <take>

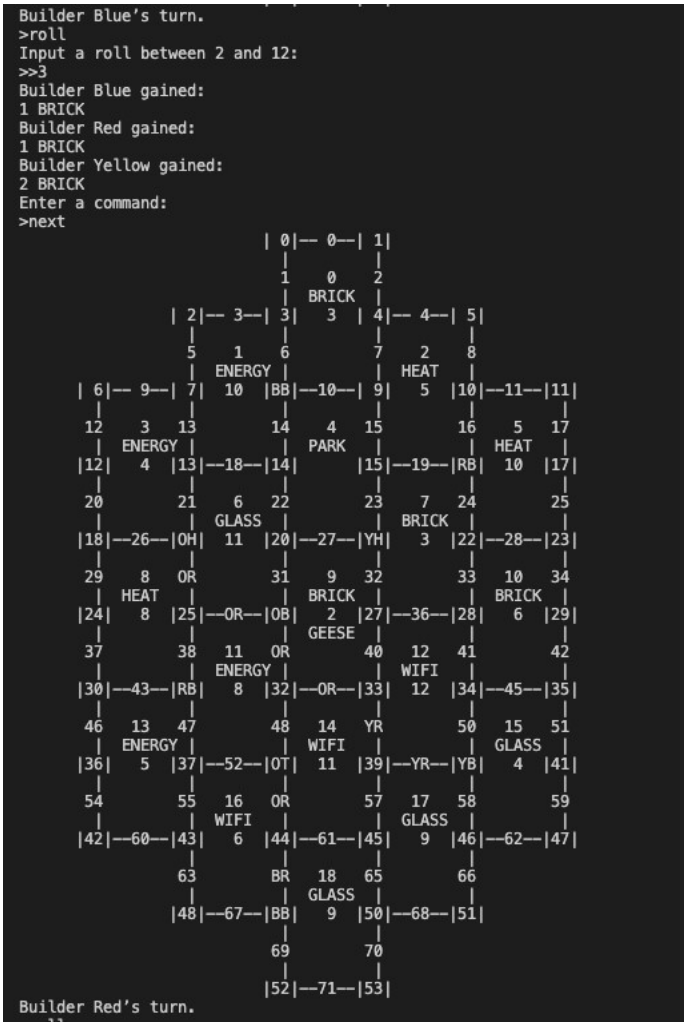
Based on the rule of this game, if a player tries to trade with <colour>, there are possible outcomes.

- You cannot trade with yourself
- <colour> does not have enough <take>
- You do not have enough <give>
- <color> accept this offer trade is successful, current player gains one <take> and loses one <give>
- <color> does not accept this offer nothing happened

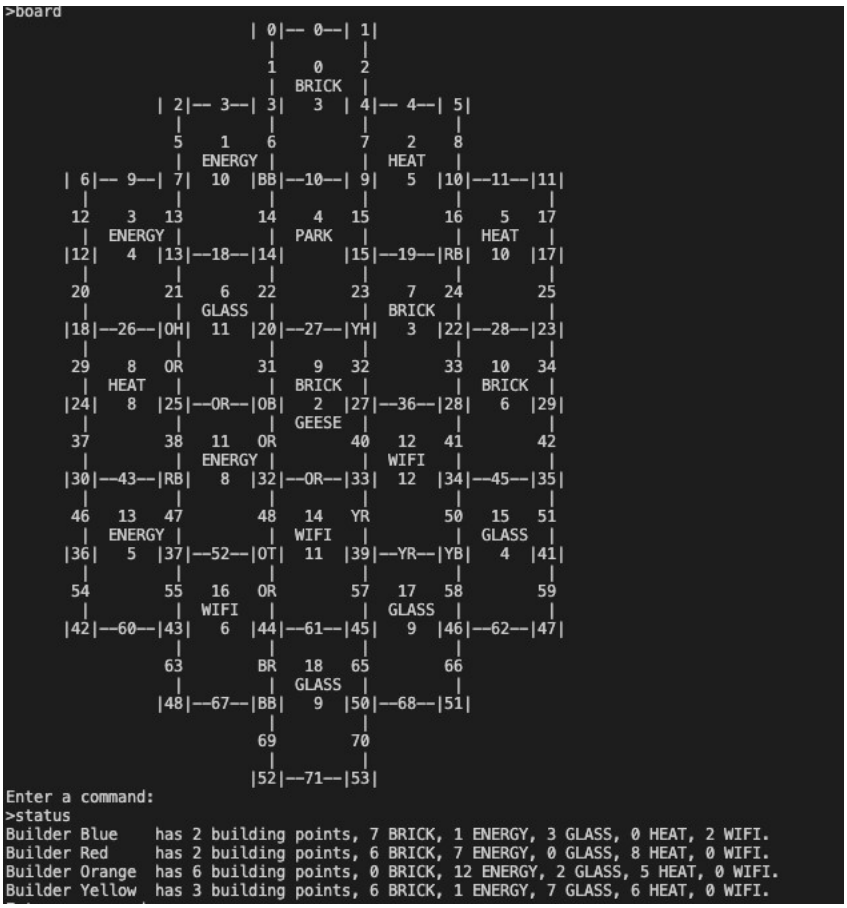


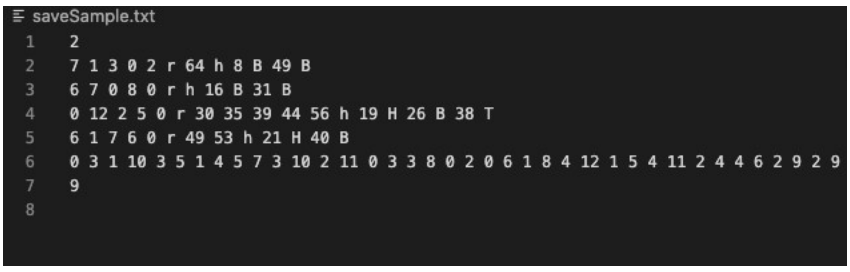
Note: When inputting <colour>, players can input the abbreviation of <colour>. (i.e. the first character of colour, both upper and lower case are valid)

“next” passes control onto the next builder in the game. It follows the sequence “Blue-Red-Orange-Yellow”.



"save <file>" saves the current game state to <file>. Here is the current information of the game.





After inputting the command, the current status will be written into the saveSample.txt

If <file> exists, then all contents will be overwritten. However, if <file> does not exist, the computer will automatically create a <file> file and then write all the information into it.

The Geese Event

Here, we use a loaded dice to trigger the Geese Event.

If 7 is rolled, Geese Event will happen.

All players who have more than ten resources in total (include 10) will lose half of their resources. And the active player can choose a tile to place the geese.

Then the current player can choose to steal resources from the player who has residences at the chosen tile. The type of stolen resources is randomly selected.

After handling the geese, the current player will complete his/her turn as normal.

```
>status
Builder Blue    has 2 building points, 4 BRICK, 1 ENERGY, 2 GLASS, 0 HEAT, 2 WIFI.
Builder Red     has 2 building points, 3 BRICK, 7 ENERGY, 0 GLASS, 8 HEAT, 0 WIFI.
Builder Orange  has 6 building points, 0 BRICK, 13 ENERGY, 1 GLASS, 5 HEAT, 0 WIFI.
Builder Yellow  has 3 building points, 0 BRICK, 0 ENERGY, 7 GLASS, 6 HEAT, 0 WIFI.
>roll
Input a roll between 2 and 12:
>>7
Builder Red Loses 9 Resources to the geese. They loss:
2 BRICK
5 Energy
2 Heat
Builder Orange Loses 9 Resources to the geese. They loss:
4 Energy
1 GLASS
4 Heat
Builder Yellow Loses 6 Resources to the geese. They loss:
2 GLASS
4 Heat
Choose where to place the GEESE.
>11
Builder Yellow can choose to steal from Red,Orange.
Choose a builder to steal from.
>R
Builder Yellow steals BRICK from builder Red.
Enter a command:
>status
Builder Blue    has 2 building points, 4 BRICK, 1 ENERGY, 2 GLASS, 0 HEAT, 2 WIFI.
Builder Red     has 2 building points, 0 BRICK, 2 ENERGY, 0 GLASS, 6 HEAT, 0 WIFI.
Builder Orange  has 6 building points, 0 BRICK, 9 ENERGY, 0 GLASS, 1 HEAT, 0 WIFI.
Builder Yellow  has 3 building points, 1 BRICK, 0 ENERGY, 5 GLASS, 2 HEAT, 0 WIFI.
Enter a command:
>
```

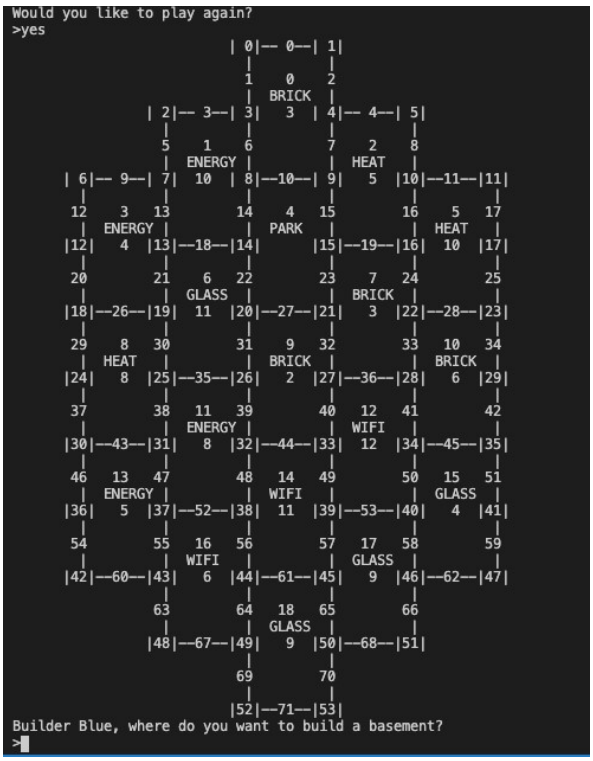
End of the Game

When a player has a total of at least 10 building pointers, he/she becomes the winner.

At this time, builders are asked whether they want to play again.

```
>status
Builder Blue    has 2 building points, 4 BRICK, 1 ENERGY, 2 GLASS, 0 HEAT, 2 WIFI.
Builder Red     has 2 building points, 3 BRICK, 7 ENERGY, 0 GLASS, 8 HEAT, 0 WIFI.
Builder Orange  has 9 building points, 2 BRICK, 10 ENERGY, 1 GLASS, 5 HEAT, 4 WIFI.
Builder Yellow  has 3 building points, 0 BRICK, 0 ENERGY, 8 GLASS, 6 HEAT, 0 WIFI.
Enter a command:
>build-res 43
>Builder Orange successfully built a Basement at 43.
Enter a command:
>status
Builder Blue    has 2 building points, 4 BRICK, 1 ENERGY, 2 GLASS, 0 HEAT, 2 WIFI.
Builder Red     has 2 building points, 3 BRICK, 7 ENERGY, 0 GLASS, 8 HEAT, 0 WIFI.
Builder Orange  has 10 building points, 1 BRICK, 9 ENERGY, 0 GLASS, 5 HEAT, 3 WIFI.
Builder Yellow  has 3 building points, 0 BRICK, 0 ENERGY, 8 GLASS, 6 HEAT, 0 WIFI.
Enter a command:
>next
Congratulations!! Orange wins!!
Would you like to play again?
```

If the response is “yes”, the game will start from the beginning.



If the answer is “no”, the game will exit.