CS 246

Object-Oriented Software Development

Final Project

Constructor

Demo

Charles Zheng (j243zhen)

Jiqi Hu (j334hu)

Kevin Xu (y574xu)

December 15, 2020

The demo will guide you through all the possible features that you will play with this game without any bonus features. To start the game, you have four ways: using a default board, using a random board, using a board with a seed, loading a board (and also the information of board and player) to the current game.

In this demo, *red words* means the command you need to enter to follow the demo. green words are describing the features or edges case that the program is performing. When enter the command, please enter them WORD-BY-WORD separated by new line.

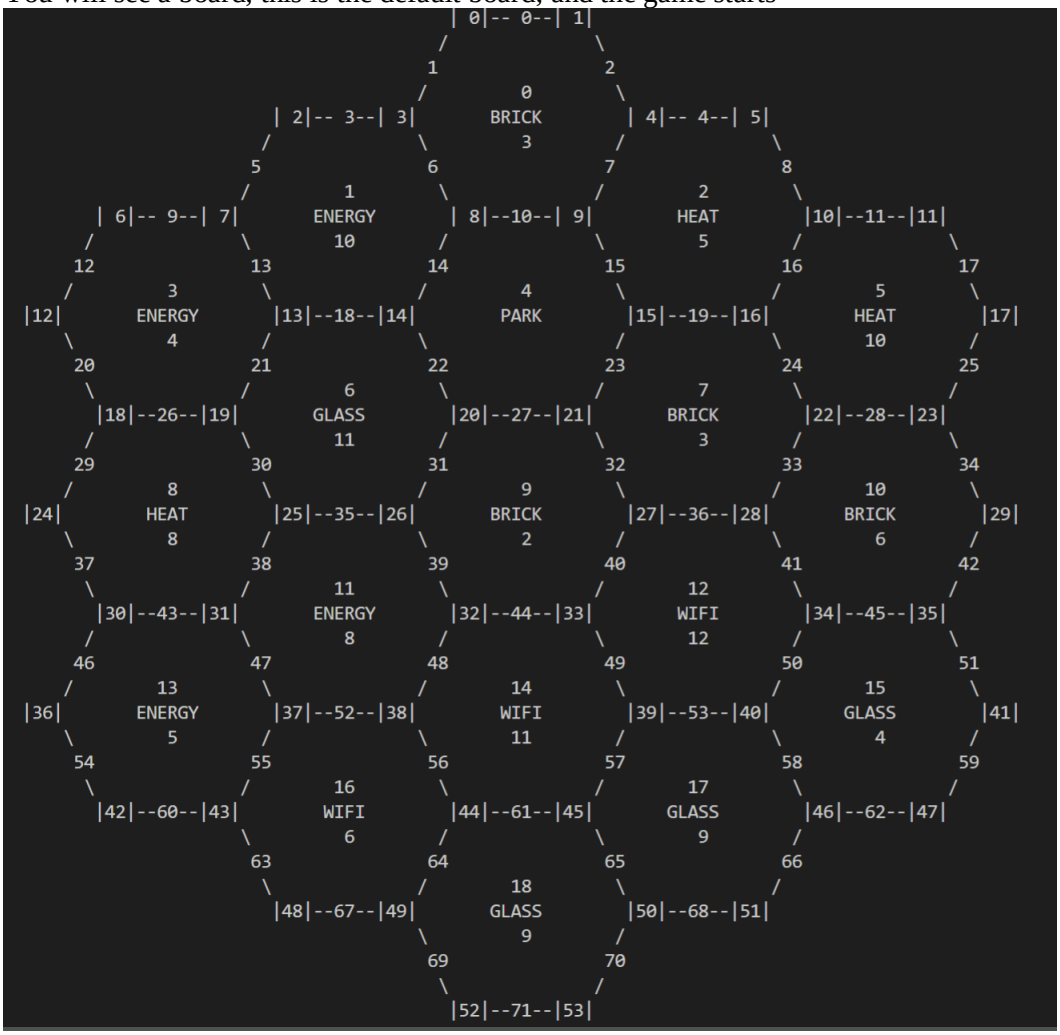
For example, please enter “trade-enter-yellow-BRICK-ENERGY” or “roll-enter-3”, instead of “trade yellow BRICK ENERGY” or “roll 3”. Thank you very much.

When testing the load command, we will provide serval files in order to test some complicated features quickly.

1. Enter the game with a default board

To start, type *./constructor*

You will see a board, this is the default board, and the game starts

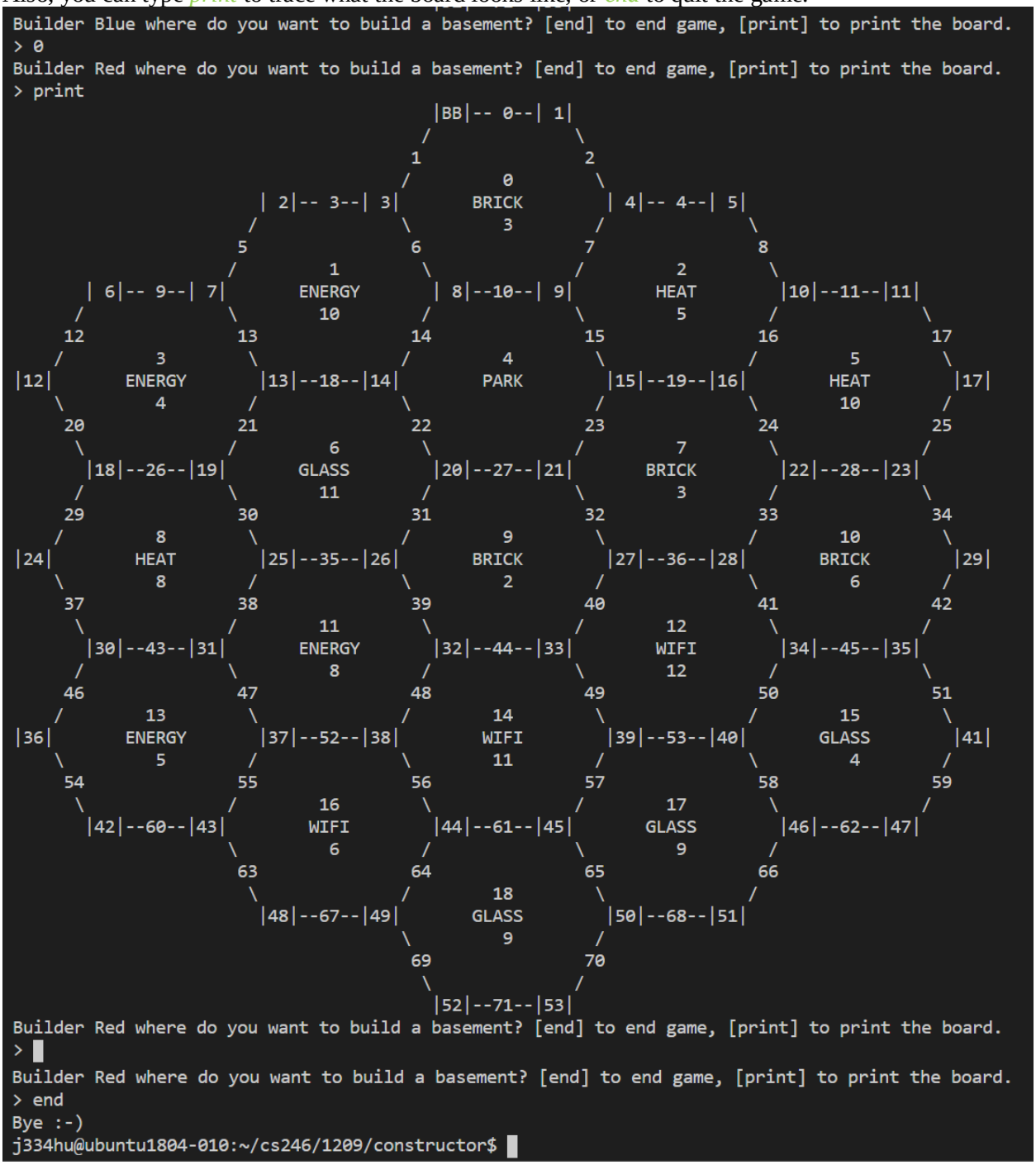


And firstly, the builders need to build house each turn in order of “blue—red—orange—yellow—yellow—orange—red—blue”

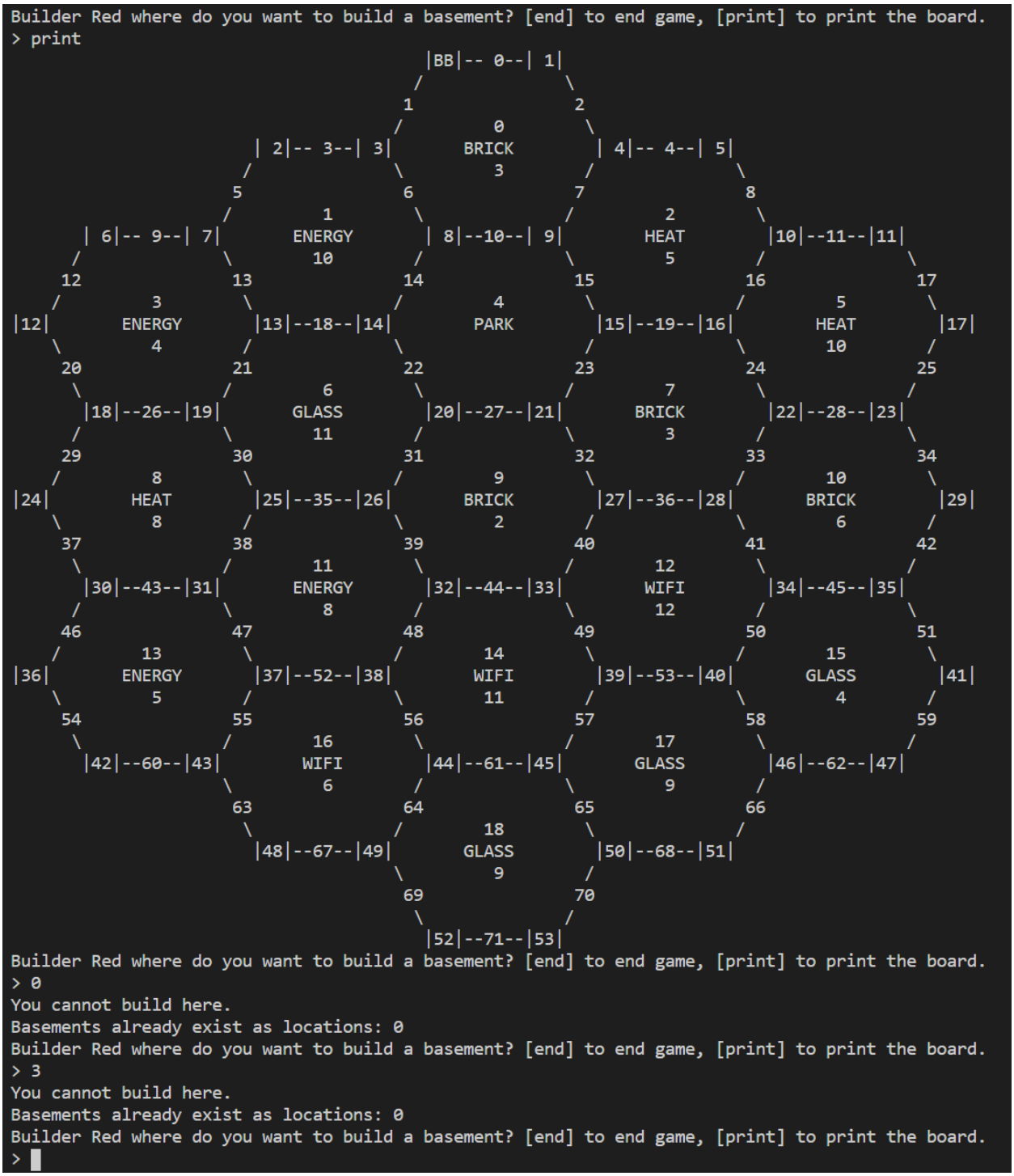
If the input is not a number, or not a valid number, the program requires player to re-enter until it’s valid. Also, it tells in which positions where buildings has been built.

```
Builder Blue where do you want to build a basement? [end] to end game, [print] to print the board.
> hi
ERROR: Builder Blue where do you want to build a basement? isn't a valid integer.
> -1
ERROR: Builder Blue where do you want to build a basement? isn't a valid integer.
> 54
You cannot build here.
Basements already exist as locations:
> 0
```

Also, you can type *print* to trace what the board looks like, or *end* to quit the game.

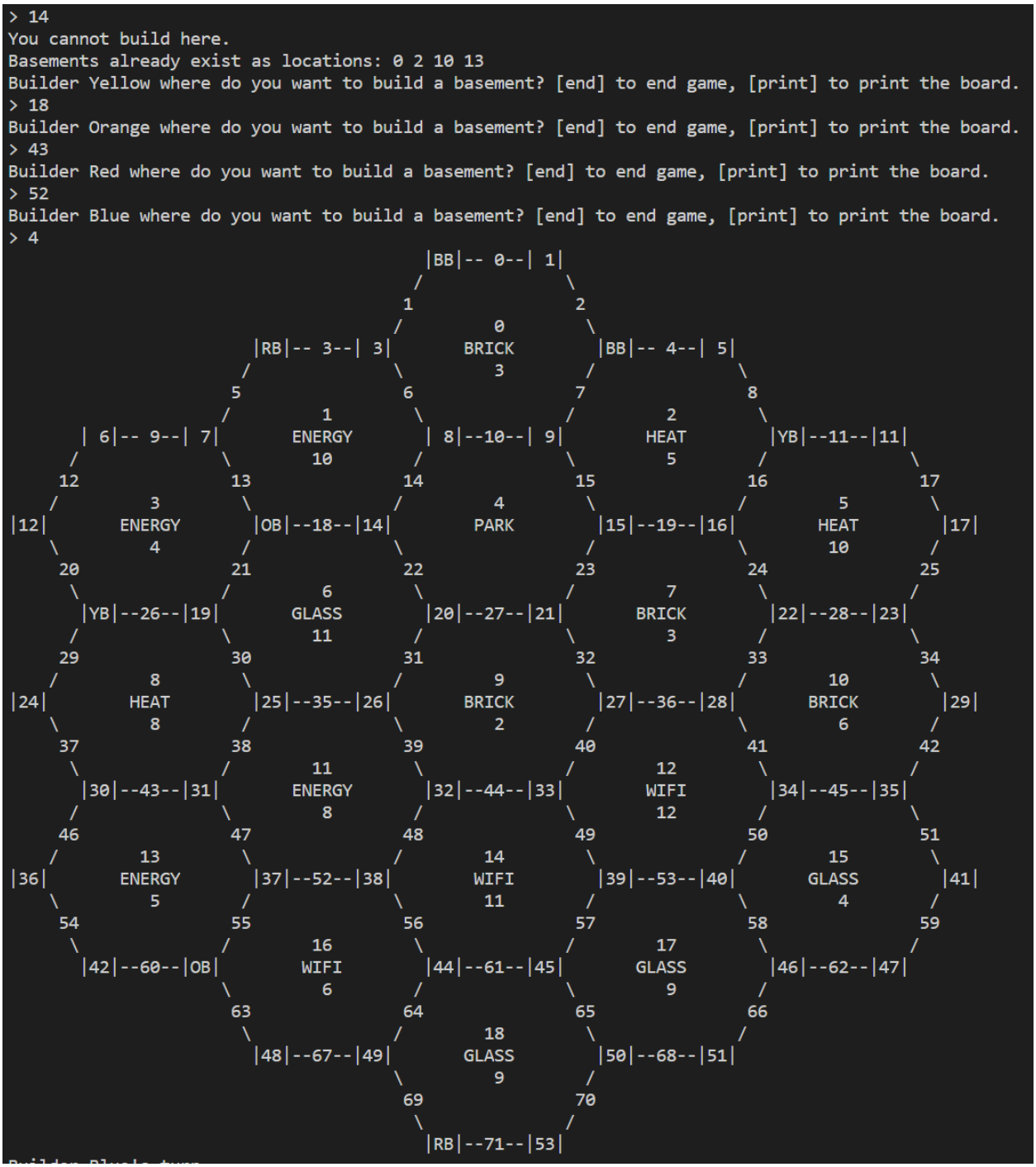


Back to placing the first two basements for builders. The input number x is invalid when there is already a basement at x, or there are some basements connect to position x:



In this case, the program will ask player to re-enter until get a valid place to build a basement.

These are the all features for the builders to place their first two basements
Assume the input are:
0 2 13 10 18 43 52 4
The graph is now



We are now in the round of "rolling dice to get resource", order of "blue—red—orange—yellow—blue…."

Type *help* to see all the command you can enter.

```
Builder Blue's turn.
> help
Valid commands:
~ load : changes current builder's dice type to 'loaded'
~ fair : changes current builder's dice type to 'fair'
~ roll : rolls the dice and distributes resources.
~ status : prints the current status of all builders in order from builder 0 to 3.
~ help : prints out the list of commands.
~ end: end the game now.
>
```

load: set the dice to loaded (which is default)
fair: set the dice to fair

```
> load
Builder Blue now has loaded Dice.
> fair
Builder Blue now has fair Dice.
> load
Builder Blue now has loaded Dice.
```

status: print the current status of all builders

```
> status
Builder Blue    has 2 building points, 0 BRICK, 0 ENERGY, 0 GLASS, 0 HEAT, 0 WIFI.
Builder Red     has 2 building points, 0 BRICK, 0 ENERGY, 0 GLASS, 0 HEAT, 0 WIFI.
Builder Orange  has 2 building points, 0 BRICK, 0 ENERGY, 0 GLASS, 0 HEAT, 0 WIFI.
Builder Yellow  has 2 building points, 0 BRICK, 0 ENERGY, 0 GLASS, 0 HEAT, 0 WIFI.
>
```

end: end the game, and automatically save the information of the game into "backup.sv" (we will come to this later)
let's roll the loaded dice: type *roll + enter*

```
> roll
Input a roll between 2 and 12:
```

Again, it will keep reading input until you enter a valid number.

Assume we roll *10*

```
Builder Blue's turn.
> roll
Input a roll between 2 and 12:
> 1
Invalid roll 1.
> -1
ERROR:  isn't a valid integer.
> what
ERROR:  isn't a valid integer.
> 13
Invalid roll 13.
> 10
The number you rolled is 10.
Builder Red gained:
1 ENERGY
Builder Orange gained:
1 ENERGY
Builder Yellow gained:
1 HEAT
Enter a command:
>
```

10 is a valid roll. And when we check the board, block 1 (ENERGY) has number 10, and since we have RB and OB in block 1, so Red and Orange each gets one ENERGY. Also block 5(HEAT) has number 11, since YB is in block 5, so Yellow gets a HEAT.

Type *next* to let yellow roll, to test another situation of rolling a loaded dice
Again we *roll* the loaded dice, this time, we roll *3* and see what happens

```
Builder Red's turn.
> roll
Input a roll between 2 and 12:
> 3
The number you rolled is 3.
Builder Blue gained:
2 BRICK
Enter a command:
>
```

There are two BB in block 0 (number 3), so Blue gains two BRICK from block 0.

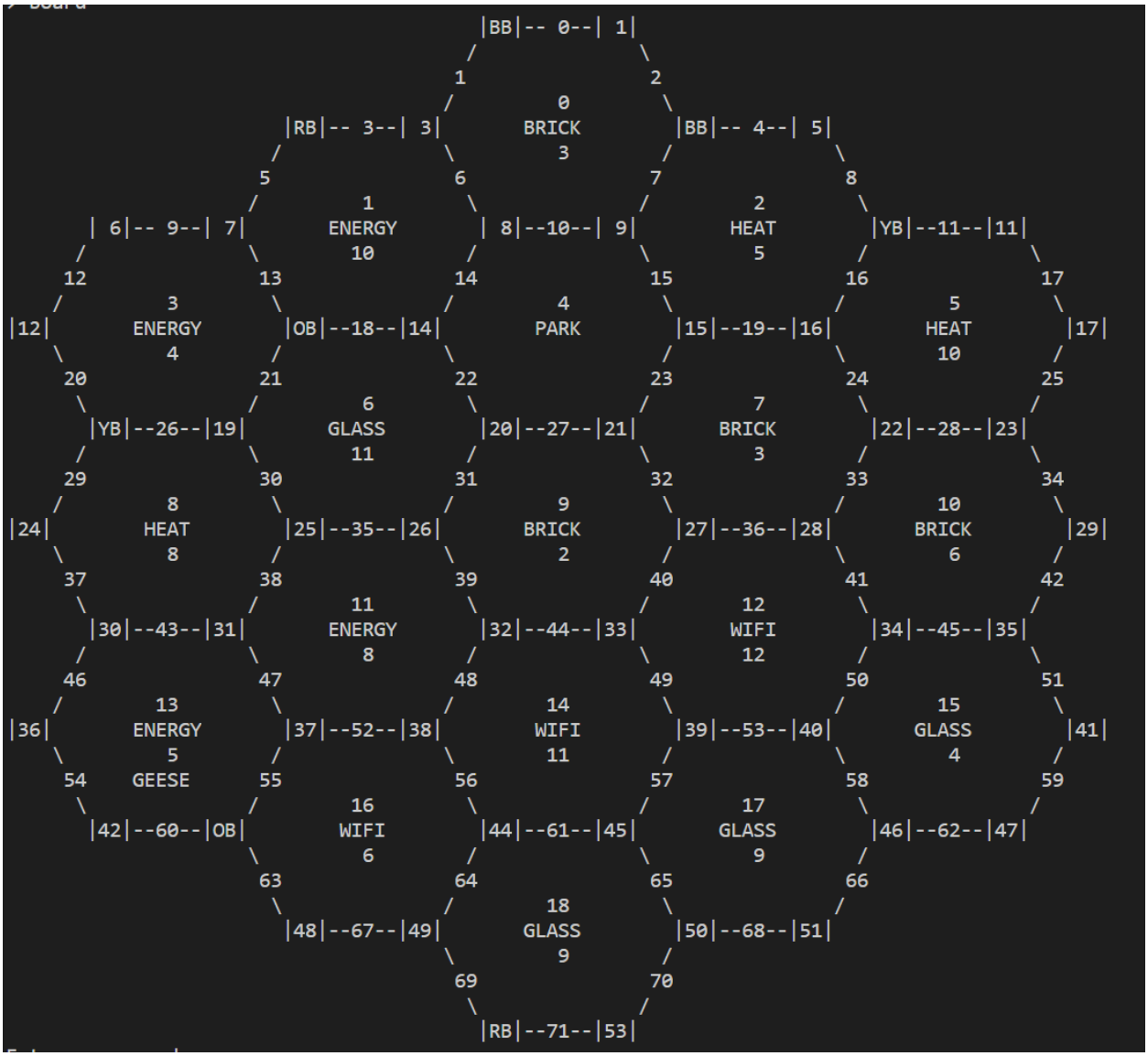
Type *next* to let yellow rolls, to test another situation of rolling a loaded dice
We now *roll 7* , and we can choose where to put geese!
(since no player has more than 10 resources, so nobody loses resource. I will test this in 4.Loading a board)

```
Builder Orange's turn.
> roll
Input a roll between 2 and 12:
> 7
The number you rolled is 7.
Choose where to place the GEESE.
>
```

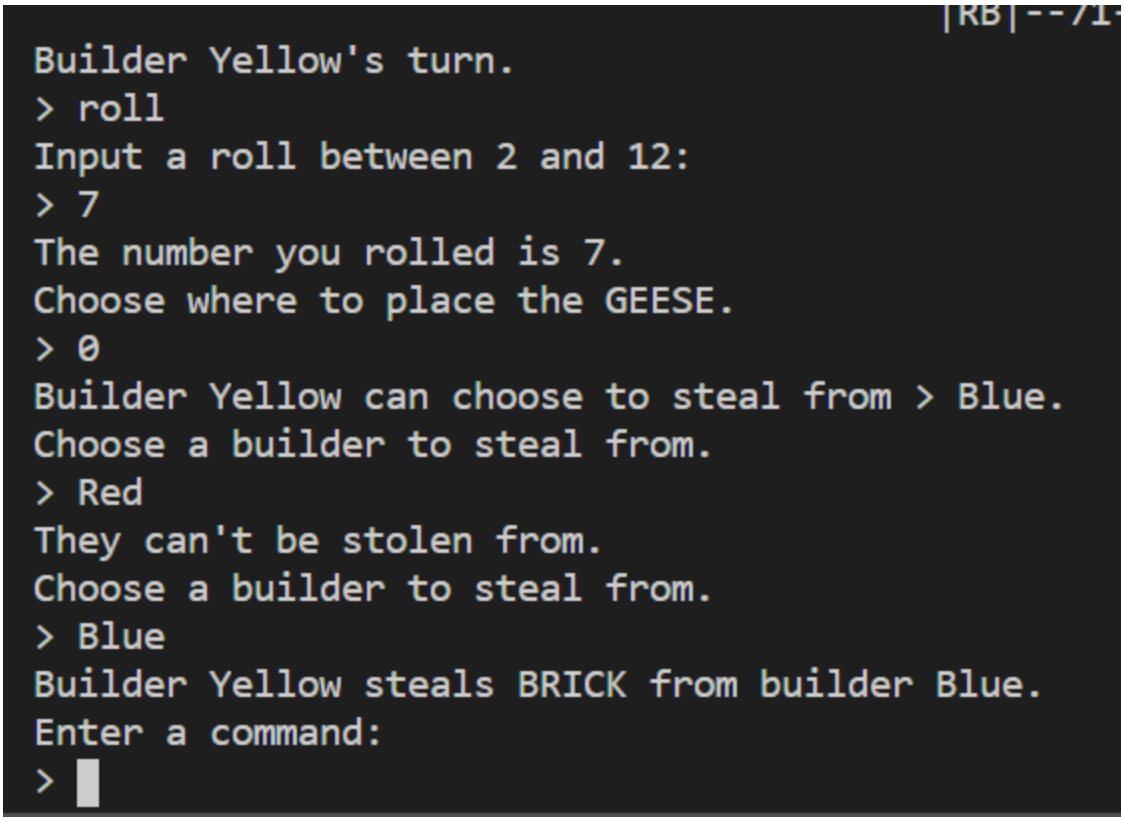
Type *13*
(Also here, it will keep asking you until you input a valid number)

```
> 7
The number you rolled is 7.
Choose where to place the GEESE.
> ?
ERROR: Choose where to place the GEESE. isn't a valid integer.
> 13
Builder Orange has no builders to steal from.
> Enter a command:
>
```

Since in block7, there is only one OB, and that YOU CANNOT STEAL YOURSELF, so orange has no one to steal.
Geese appears in the board (type *board*):



Type *next* to test another situation
Now it's Yellow's turn, *roll + enter + 7*, and place the geese at *0*
Type *Blue*



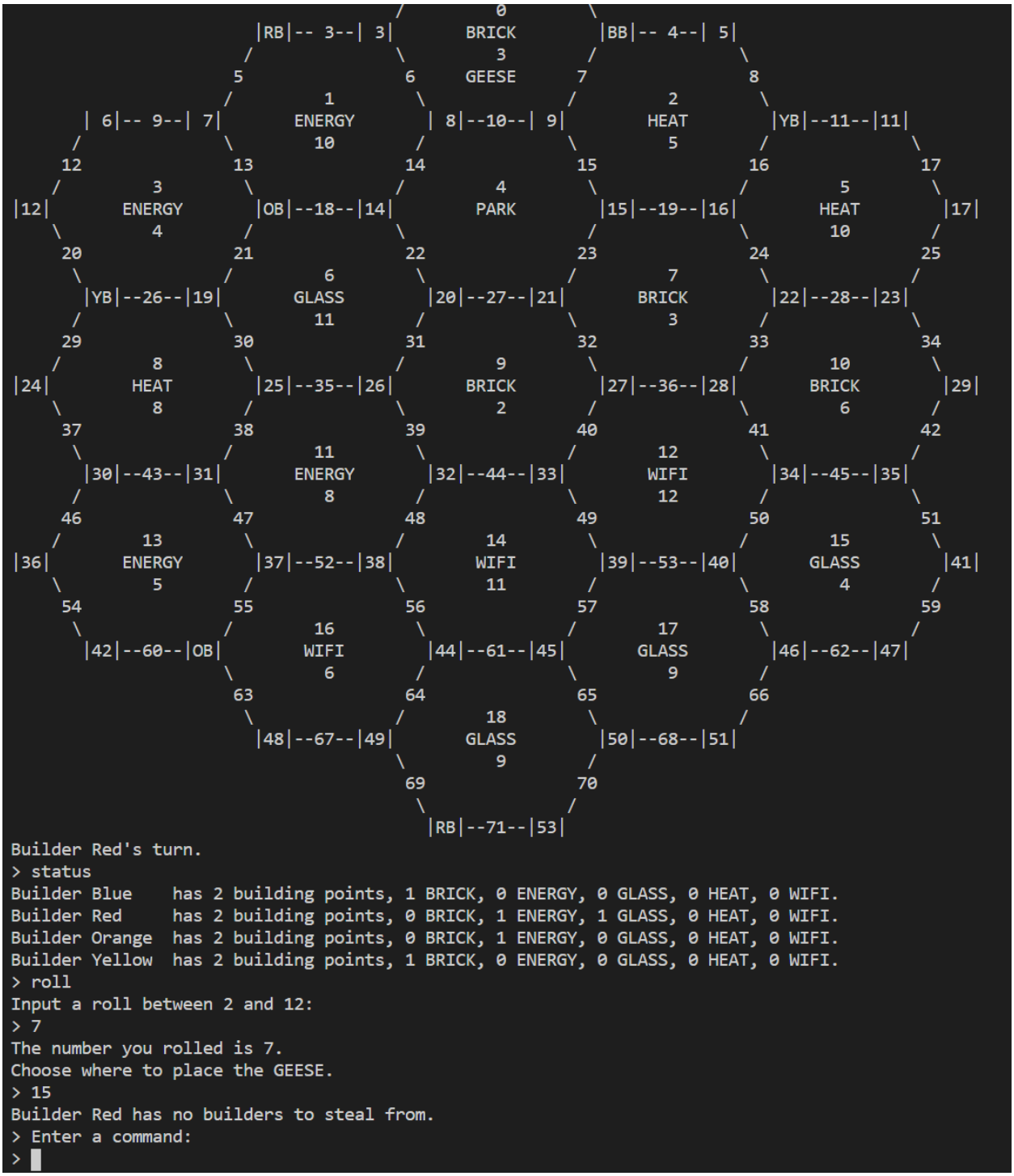
Since there are two BB in block 0, so it you can ONLY steal from blue. When asking who to steal from, you have to enter a valid color to steal. Since so far blue has only 2 BRICK, so it automatically steals one BRICK from blue.

Type *next* to test another situation

Here we *roll + enter + 9* to give red some resource for testing. And enter *next*.

Type *status* to see the current status.

Type *roll + enter + 7* to put the geese at position *15*



Block 15 has no buildings. So for sure Red has no builder to steal from
Type *next* to test another situation
Type *roll + enter + 7* to put the geese at position *1,* steals from *Red*
Type *next* to test another situation

Type *roll + enter + 7* to put the geese at position *1*

```
> roll
Input a roll between 2 and 12:
> 7
The number you rolled is 7.
Choose where to place the GEESE.
> 1
Geese can't move here.
>
```

Since we have put geese at 1 last turn, and we CANNOT put geese at the same place in two continuous turn of rolling 7, so it requires us to put another place. We type *15* and steals from no one.
Type *next* to test another situation

Type *roll + enter + 7* to put the geese at position *1*
This time, it works

```
> 1
Builder Blue can choose to steal from > Red, Orange.
Choose a builder to steal from.
> orange
Builder Blue steals GLASS from builder Orange.
Enter a command:
> status
Builder Blue    has 2 building points, 1 BRICK, 0 ENERGY, 1 GLASS, 0 HEAT, 0 WIFI.
Builder Red     has 2 building points, 0 BRICK, 1 ENERGY, 0 GLASS, 0 HEAT, 0 WIFI.
Builder Orange  has 2 building points, 0 BRICK, 1 ENERGY, 0 GLASS, 0 HEAT, 0 WIFI.
Builder Yellow  has 2 building points, 1 BRICK, 0 ENERGY, 0 GLASS, 0 HEAT, 0 WIFI.
Enter a command:
>
```

We type *Orange*, since Orange has one ENERGY and one GLASS, it randomly steals one resource (GLASS) from orange. Type *status* to check that orange does lose one GLASS and blue does gains one GLASS.
Type *next*
We test fair dice here. Type *fair + roll*

```
Builder Red's turn.
> fair
Builder Red now has fair Dice.
> roll
The number you rolled is 12.
No builders gained resources.
Enter a command:
>
```

Fair dice randomly gives you a valid number (12) and dice number 12 doesn't give any one resources.

So far, we have finish testing all the features in "rolling dice turn" (except geese randomly steals resources from player who has more than 10 resources).

Next, we go to the next turn "operation turn". Type *help* to see all operations

```
Enter a command:
> 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>.
~ market <sell> <buy> : attempts to sell resources on the market, giving four resource of type <sell> and receiving one resource of type <buy>.
~ 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.
~ end : end the game.
Enter a command:
>
```

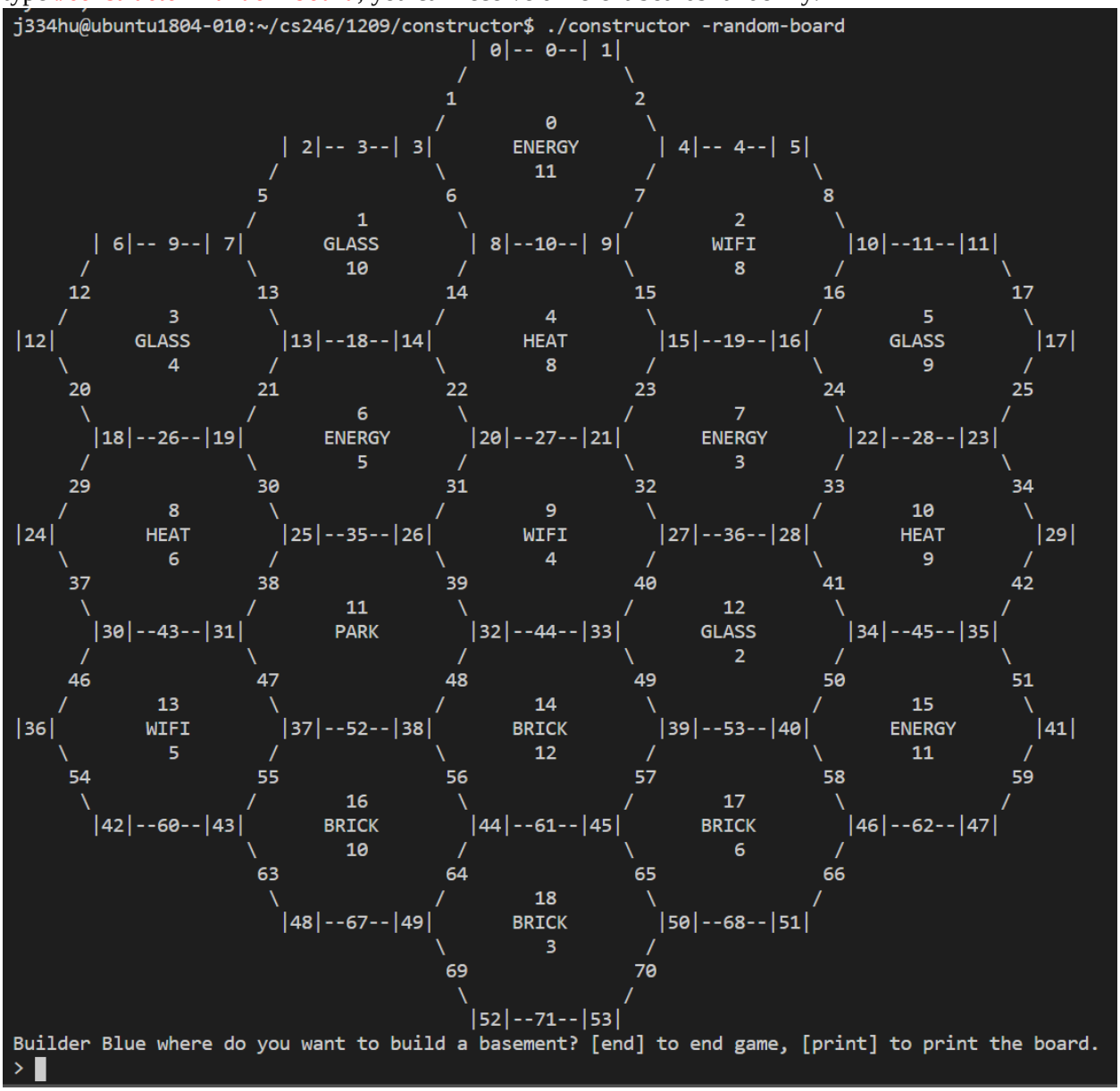
We have tested board, which just simply print the current board.
We have tested status, which presents the current status.
We have tested next, which goes to the next person.
end terminates the game, and save the game into back.sv
We will test (residences, build-road, build-res, improve, trade, market, save) in 4.Loading a board. They have the same function in all 4 types of game modes. For your convenience, loading a board is easier for testing those features.
Type *end* and we goes to the next mode.
You may check backup.sv, it becomes

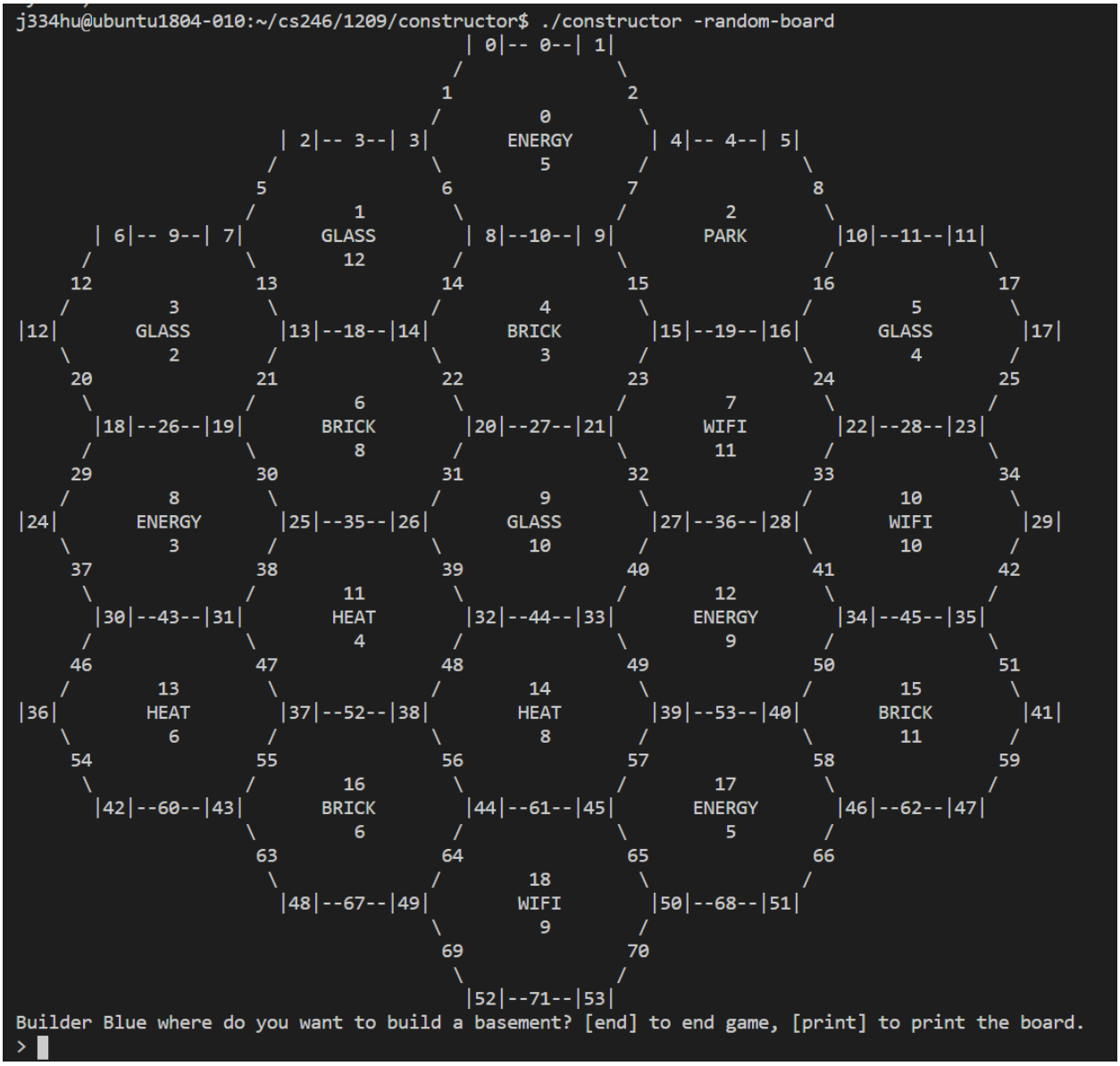
```
9
1 0 1 0 0 r h 0 B 4 B
0 1 0 0 0 r h 2 B 52 B
0 1 0 0 0 r h 13 B 43 B
1 0 0 0 0 r h 10 B 18 B
0 3 1 10 3 5 1 4 5 7 3 10 2 11 0 3 3 8 0 2 0 6 1 8 4 12 1 5 4 11 2 4 4 6 2 9 2 9
1
```

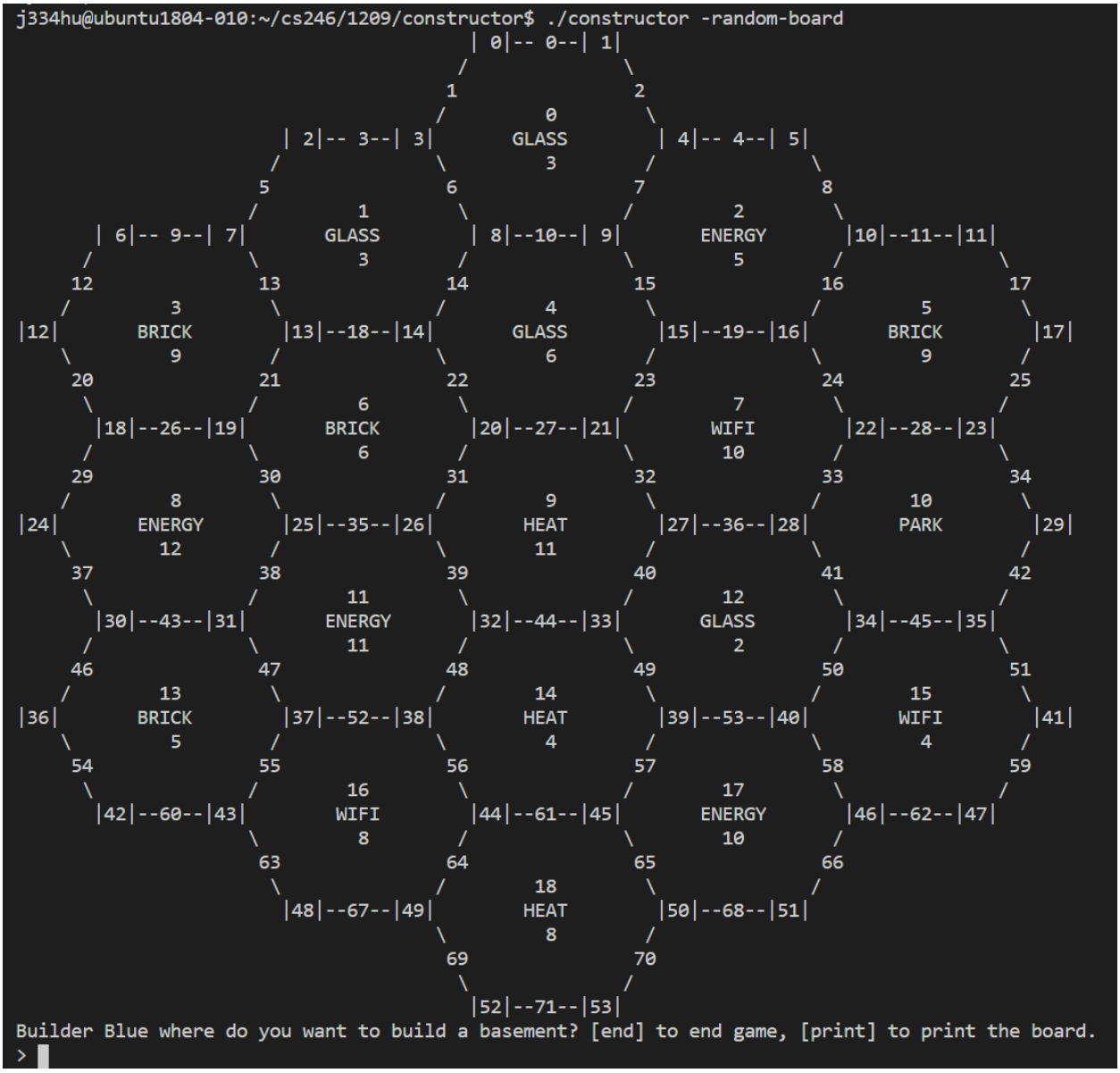
This is the information of the game that we just played.

2. Enter the game with a random board

type *./constructor -random-board*, you can receive different boards randomly.




j334hu@ubuntu1804-010:~/cs246/1209/constructor$ ./constructor -random-board
Builder Blue where do you want to build a basement? [end] to end game, [print] to print the board.
>

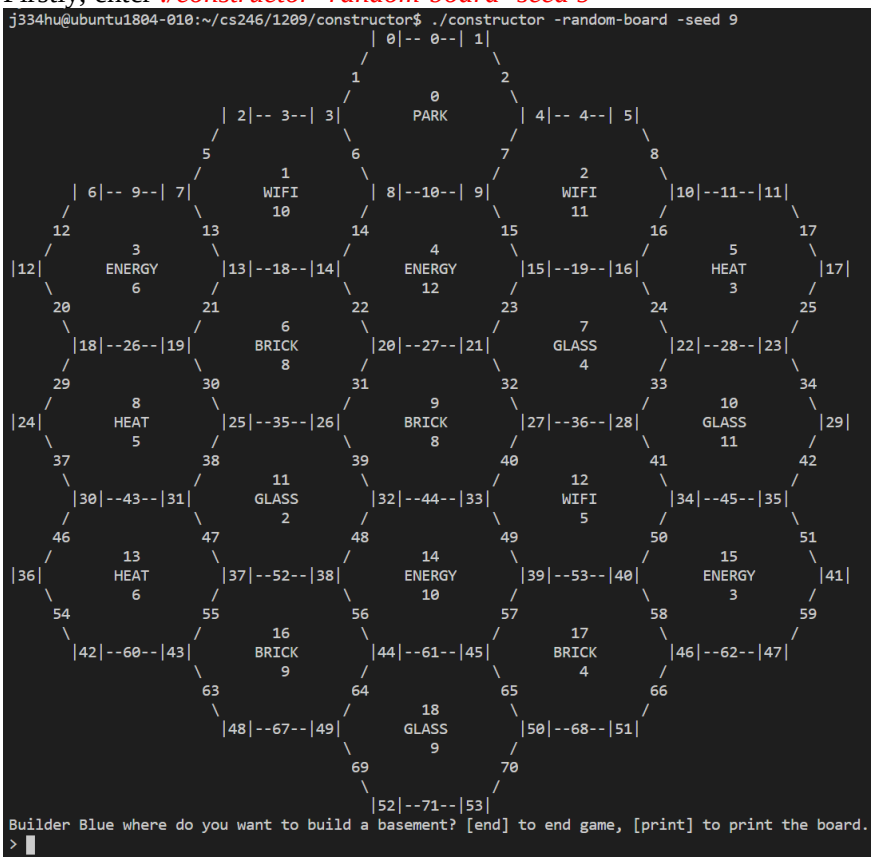


As you can see, those boards are randomly presented (random distribution of resources and dice-number of each block)

You may play your game normally at any random board.

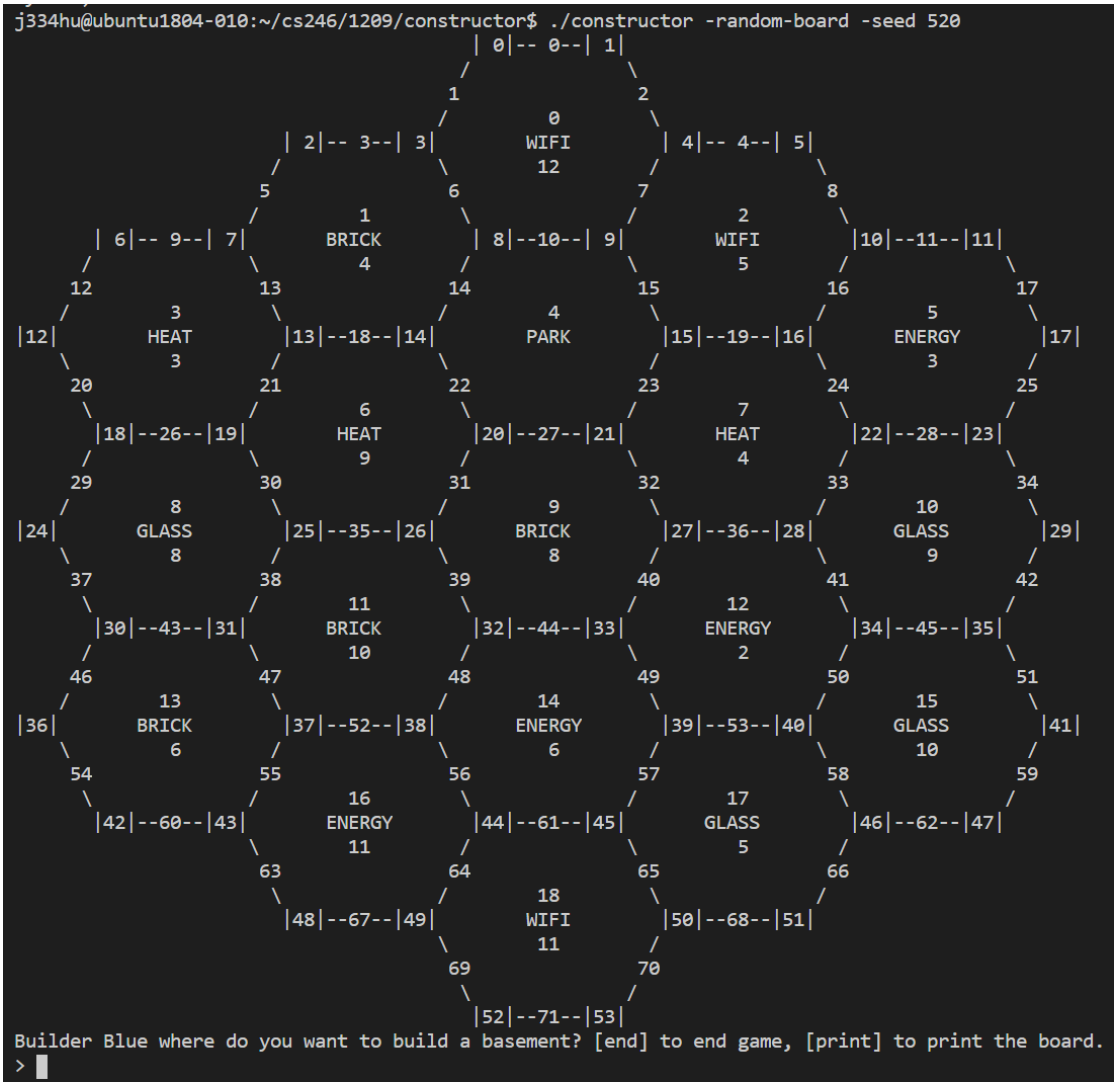
3. Enter the game with a seed

Firstly, enter *./constructor -random-board -seed 9*



You receive a random board that is BASED ON your seed.

i.e. if you try another seed (such as 520) *./constructor -random-board -seed 520*, you get a different board



Same seed will give same board all the time, also, seed will change the randomness of the game! i.e. if this game is seed 9, and we roll the first fair dice to get 2, then, every time we use this seed to play this game, using the same command, and the first fair dice we roll will be 2 because the seed will control the randomness of the game.

4. Loading a board

to load a game, we need to prepare a file with valid information.
Let's name this file test1.txt.

```
9
1 2 3 4 5 r h 0 B 15 B
0 0 0 0 0reside r 21 30 h 2 B 13 B 25 H
10 10 10 10 10 r 16 11 17 19 h 4 B 10 B 17 H
2 2 2 2 3 r h 6 B 8 B
3 11 0 11 2 12 2 8 5 7 2 3 2 5 1 9 3 9 3 4 4 5 1 10 4 2 0 10 4 6 1 6 0 3 0 8 1 4
8

```

Enter *./constructor -load test1.txt*
Since term = 9, 9 % 4 = 1, so it's red's term.
Enter *roll 9*

```
Builder Red's turn.
> roll
Input a roll between 2 and 12:
> 9
The number you rolled is 9.
Builder Blue gained:
1 ENERGY
Enter a command:
>
```

Since geese is in block 8, so block 8 doesn't provide any resource even if 9 is rolled. Block 7 gives one ENERGY to blue.
Enter *board* and *status* to check that it satisfies test1.txt

Type *residences*

```
> residences
Red has built:
2 B
13 B
25 H
Enter a command:
>
```

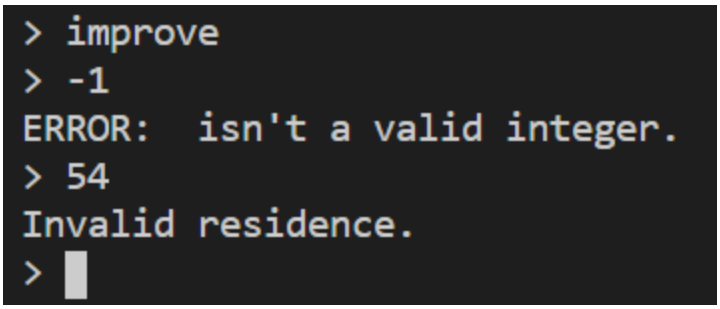
This is also correct
Enter *next*
Enter *roll 7*

```
                                                    |52|--71--|53|
Builder Orange's turn.
> roll
Input a roll between 2 and 12:
> 7
The number you rolled is 7.
Builder Blue loses 8 resources to the geese. They lose :
2 ENERGY
3 GLASS
3 WIFI
Builder Orange loses 25 resources to the geese. They lose :
8 BRICK
1 ENERGY
6 GLASS
6 HEAT
4 WIFI
Builder Yellow loses 5 resources to the geese. They lose :
1 BRICK
1 GLASS
2 HEAT
1 WIFI
Choose where to place the GEESE.
> 18
Builder Orange has no builders to steal from.
> Enter a command:
>
```

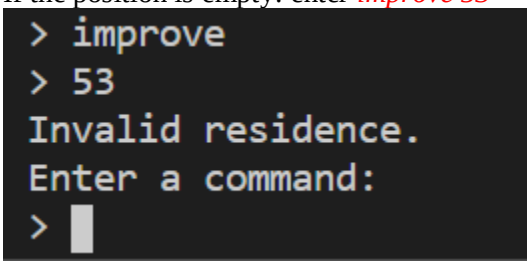
Since blue has 16 (>=10) resources, blue randomly loses 8 (16/2) resources to geese.
Since orange has 50 (>=10) resources, orange randomly loses 25 (50/2) resources to geese.
Since yellow has 11 (>=10) resources, yellow randomly loses 5 (11/2) resources to geese.
Put geese at *18*
Check status:
Enter *status* (notice: since the resources are randomly lost, so status may be different when you are testing here)

```
> status
Builder Blue    has 2 building points, 1 BRICK, 1 ENERGY, 0 GLASS, 4 HEAT, 2 WIFI.
Builder Red     has 4 building points, 1 BRICK, 1 ENERGY, 1 GLASS, 0 HEAT, 1 WIFI.
Builder Orange  has 4 building points, 2 BRICK, 9 ENERGY, 4 GLASS, 4 HEAT, 6 WIFI.
Builder Yellow  has 2 building points, 1 BRICK, 2 ENERGY, 1 GLASS, 0 HEAT, 2 WIFI.
Enter a command:
>
```

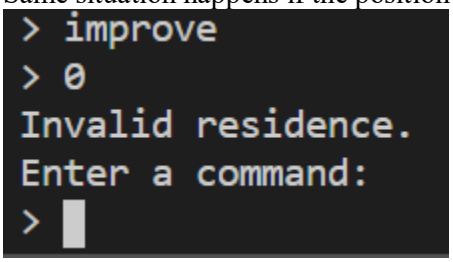
It changes.
Now let's test improve
Enter *improve*
Again, it needs valid input: number between 0 and 53 inclusive



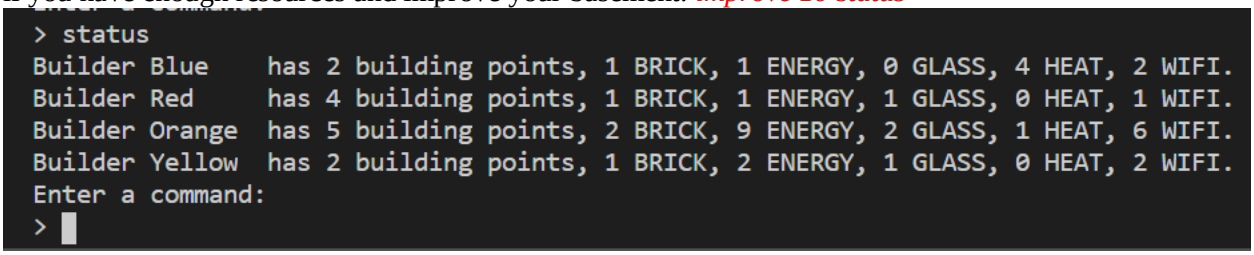
If the position is empty: enter *improve 53*



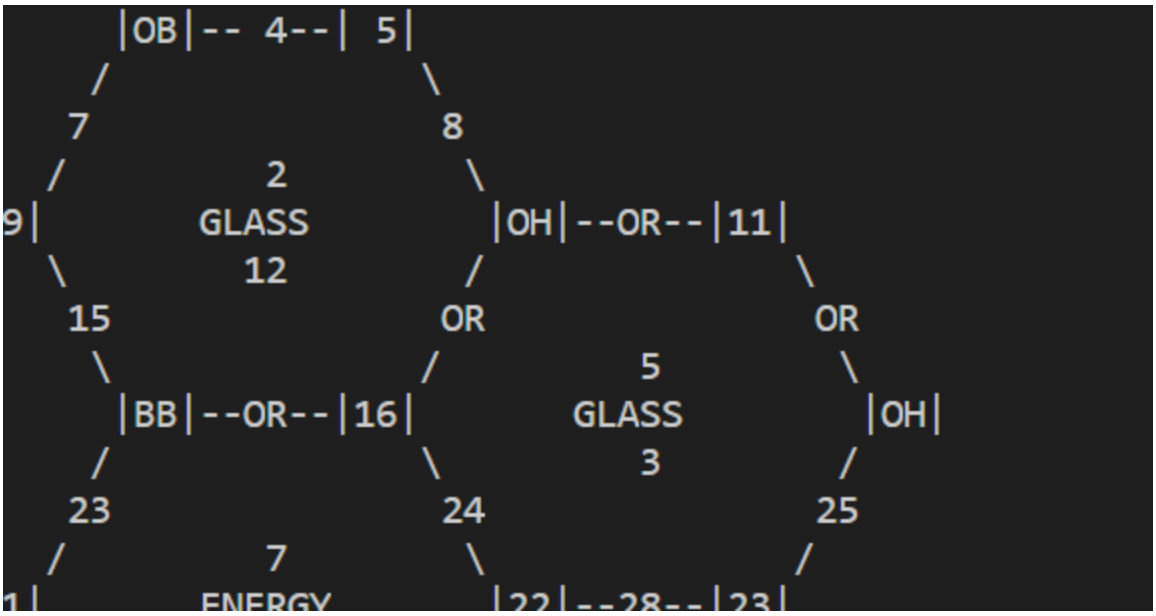
Same situation happens if the position has other player's building: *improve 0*



if you have enough resources and improve your basement: *improve 10 status*



Enter *board*



Status and board have been updated.
Type *end*
Using test1.txt again: *./constructor -load test1.txt*
Enter: *roll 4*
If you don't have enough resource, you may not improve anything
Enter: *improve 2*: (try to improve a basement)

```
Enter a command:
> improve
> 2
> 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.
Enter a command:
>
```

Enter : *improve 25* (try to improve a house)

```
Enter a command:
> improve
> 25
> 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.
Enter a command:
>
```

Enter *next roll 4*
Enter *improve 17* (successfully improve a house to tower)
You may enter *board, status, residences* to see the data being updated.
Enter *save res1.txt*, then you can see the game saved in the file res1.txt that you may continue playing in the future by entering ./constructor -load res1.txt
End *end*

Testing build-res + build-road
Using file test2.txt:

test2.txt

```
9
1 2 3 4 5 r h 0 B 15 B
1 1 1 0 0 r 21 30 26 h 2 B 13 B 25 H
100 100 100 100 100 r 16 70 71 11 17 28 25 19 h 4 B 10 B 17 H 50 T 52 H
2 2 2 2 3 r h 6 B 8 B
3 11 0 11 2 12 2 8 5 7 2 3 2 5 1 9 3 9 3 4 4 5 1 10 4 2 0 10 4 6 1 6 0 3 0 8 1 4
8
```

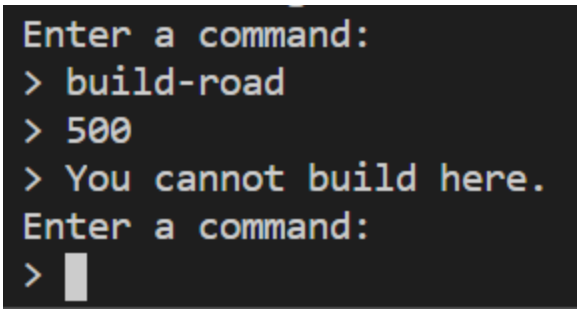
Enter: *./constructor -load test2.txt*
Enter: *roll 2*
When doesn't have enough resource but trying to build-road: *build-road 11 build-road 18*

```
Enter a command:
> build-road
> 11
You do not have enough resources.
The cost of a Road is one HEAT and one WIFI resource.
> You cannot build here.
Enter a command:
> build-road
> 18
You do not have enough resources.
The cost of a Road is one HEAT and one WIFI resource.
> You cannot build here.
Enter a command:
>
```

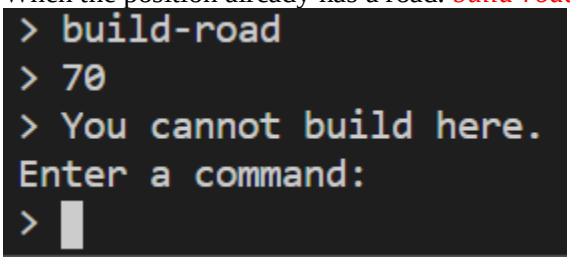
When doesn't have enough resource but trying to build-res: *build-res 18*

```
Enter a command:
> build-res
> 2
You do not have enough resources.
The cost of a Basement is one BRICK, one ENERGY, one GLASS, and one WIFI resource.
Enter a command:
>
```

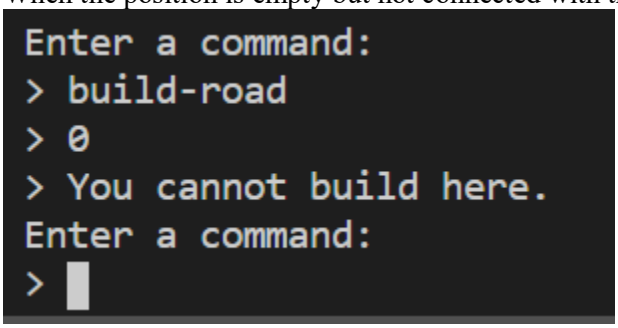
Enter: *next roll 2*
When trying to build-road at invalid position: *build-road 500*



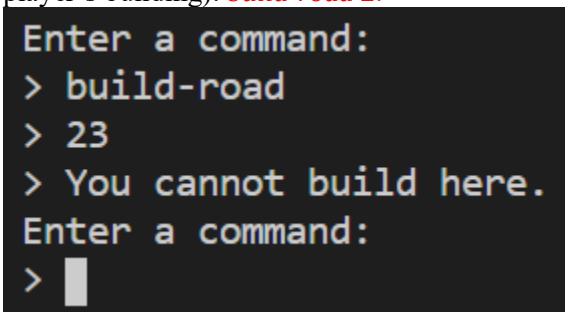
When the position already has a road: *build-road 70*



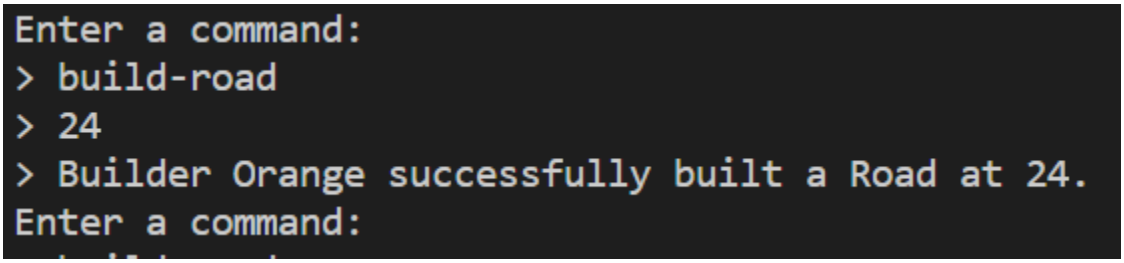
When the position is empty but not connected with the current player's building: *build-road 0*



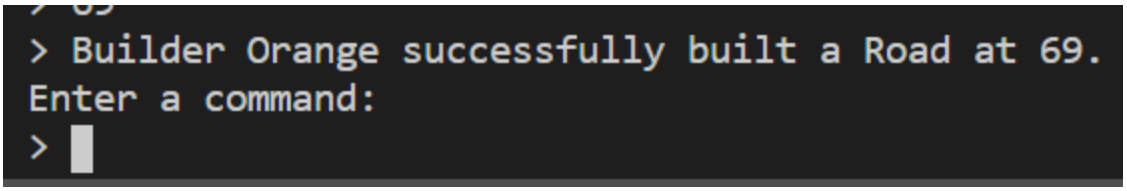
When try to build a road (connected to the current player's road but also connect to a another player's building): *build-road 27*



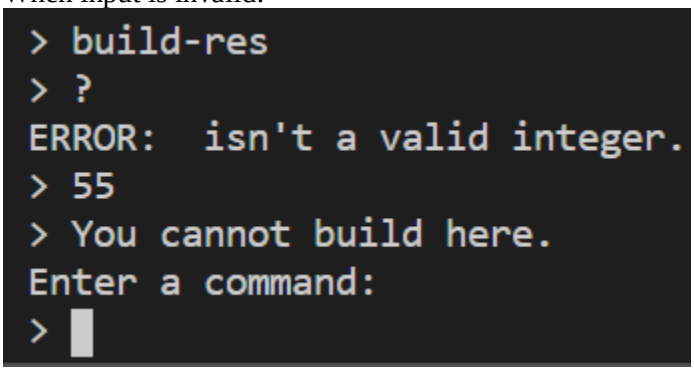
When successfully build a road at a empty place that is connect to the current player's another road: *build-road 24*



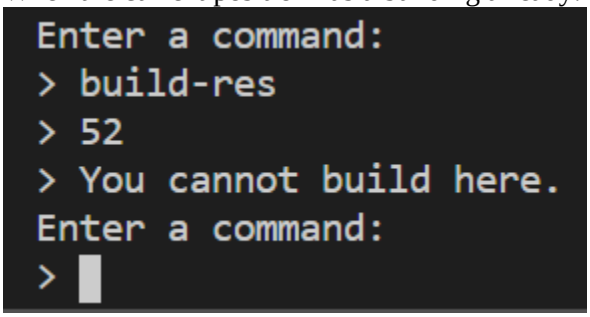
When successfully build a road at a empty place that is connect to the current player's building: *build-road 69*



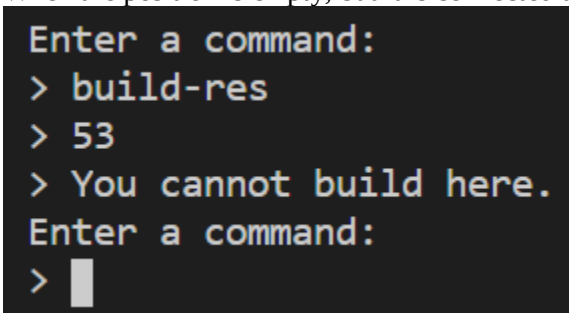
You may use *board, status* to check that the data has been updated after building roads.
Now test build-res
When input is invalid:



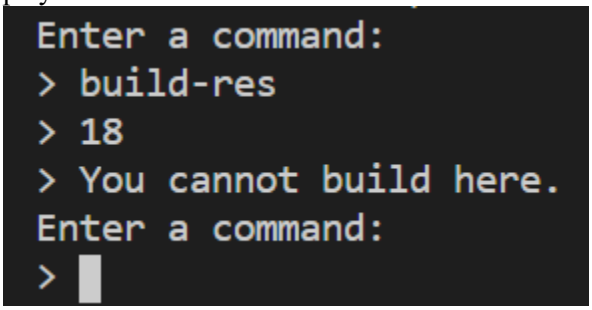
When the current position has a building already: *build-res 52*



When the position is empty, but it is connected to other buildings: *build-res 53*



When the position is empty, not connected to any building, but not connected to the current player's road: *build-res 18*



When successfully build a basement: *build-res 22*

```
> build-res
> 22
> Builder Orange successfully built a Basement at 22.
Would you like to play again?
> 
```

After building a basement at 22, orange's points increases from 9 to 10, the game end. Then you may choose to play again (yes) or quit (no).

If you want to play again: enter *yes*

```
Would you like to play again?
> yes
                                    | 0|-- 0--| 1|
                                   /             \
                                  1               2
                                 /        0        \
                     | 2|-- 3--| 3|       HEAT       | 4|-- 4--| 5|
                    /             \       11       /             \
                   5               6               7               8
                  /        1       \             /        2       \
      | 6|-- 9--| 7|      BRICK      | 8|--10--| 9|      GLASS      |10|--11--|11|
     /              \       11       /             \       12       /              \
    12              13              14              15              16              17
   /        3       \             /        4       \             /        5       \
|12|      GLASS      |13|--18--|14|       PARK       |15|--19--|16|      GLASS      |17|
   \        8       /             \                 /             \        3       /
    20              21              22              23              24              25
     \             /        6       \             /        7       \             /
      |18|--26--|19|      GLASS      |20|--27--|21|      ENERGY     |22|--28--|23|
     /              \        5       /             \        9       /              \
    29              30              31              32              33              34
   /        8       \             /        9       \             /       10       \
|24|      HEAT       |25|--35--|26|       HEAT       |27|--36--|28|       WIFI      |29|
   \        9       /             \        4       /             \        5       /
    37              38              39              40              41              42
     \             /       11       \             /       12       \             /
      |30|--43--|31|      ENERGY     |32|--44--|33|       WIFI      |34|--45--|35|
     /              \       10       /             \        2       /              \
    46              47              48              49              50              51
   /       13       \             /       14        \             /       15       \
|36|      BRICK      |37|--52--|38|       WIFI       |39|--53--|40|      ENERGY     |41|
   \       10       /             \        6        /             \        6       /
    54              55              56              57              58              59
     \             /       16       \             /       17       \             /
      |42|--60--|43|      BRICK      |44|--61--|45|      BRICK      |46|--62--|47|
             \       3       /             \       8       /
              63              64              65              66
               \             /       18        \             /
                |48|--67--|49|      ENERGY      |50|--68--|51|
                             \        4        /
                              69              70
                               \             /
                                |52|--71--|53|
Builder Blue where do you want to build a basement? [end] to end game, [print] to print the board.
> 
```

It uses the same graph, start from the beginning.

Now let's test trade and market
Enter *./constructor -load test2.txt*
Enter *roll 2 status*

```
Builder Red's turn.
> roll
Input a roll between 2 and 12:
> 2
The number you rolled is 2.
No builders gained resources.
Enter a command:
> status
Builder Blue    has 2 building points, 1 BRICK, 2 ENERGY, 3 GLASS, 4 HEAT, 5 WIFI.
Builder Red     has 4 building points, 1 BRICK, 1 ENERGY, 1 GLASS, 0 HEAT, 0 WIFI.
Builder Orange  has 9 building points, 100 BRICK, 100 ENERGY, 100 GLASS, 100 HEAT, 100 WIFI.
Builder Yellow  has 2 building points, 2 BRICK, 2 ENERGY, 2 GLASS, 2 HEAT, 3 WIFI.
Enter a command:
>
```

You cannot trade with yourself: *trade Red ENERGY HEAT*

```
Enter a command:
> trade
> Red
> ENERGY
> HEAT
> Can't trade with yourself.
Enter a command:
>
```

If you don't have enough resource but trade with others: *trade Orange WIFI BRICK*

```
Enter a command:
> trade
> Orange
> WIFI
> BRICK
> You don't have enough ENERGY.
Enter a command:
>
```

Enter *next roll 2 status*
When you have enough resource but the people you are trading don't have the resource that you want: *trade Red GLASS HEAT*

```
> tarde
Invalid command.
Please enter 'help' for a list of valid commands.
Enter a command:
> trade
> Red
> GLASS
> HEAT
> Red doesn't have enough HEAT.
Enter a command:
>
```

Also, do not trade the same resource: *trade Red BRICK BRICK*

```
Enter a command:
> trade
> Red
> BRICK
> BRICK
> Why are you trading for the same resource...
Enter a command:
>
```

If the trade is valid: *trade Red WIFI BRICK*
And Red refuse to trade: *no*
Status remain same: *status*

```
Enter a command:
> status
Builder Blue    has 2 building points, 1 BRICK, 2 ENERGY, 3 GLASS, 4 HEAT, 5 WIFI.
Builder Red     has 4 building points, 1 BRICK, 1 ENERGY, 1 GLASS, 0 HEAT, 0 WIFI.
Builder Orange  has 9 building points, 100 BRICK, 100 ENERGY, 100 GLASS, 100 HEAT, 100 WIFI.
Builder Yellow  has 2 building points, 2 BRICK, 2 ENERGY, 2 GLASS, 2 HEAT, 3 WIFI.
Enter a command:
> trade
> Red
> WIFI
> BRICK
> > > Orange offers Red one WIFI for one BRICK.
Does Red accept this offer?
> no
Red declined the trade.
Enter a command:
> status
Builder Blue    has 2 building points, 1 BRICK, 2 ENERGY, 3 GLASS, 4 HEAT, 5 WIFI.
Builder Red     has 4 building points, 1 BRICK, 1 ENERGY, 1 GLASS, 0 HEAT, 0 WIFI.
Builder Orange  has 9 building points, 100 BRICK, 100 ENERGY, 100 GLASS, 100 HEAT, 100 WIFI.
Builder Yellow  has 2 building points, 2 BRICK, 2 ENERGY, 2 GLASS, 2 HEAT, 3 WIFI.
Enter a command:
>
```

If the trade is valid: *trade Red WIFI BRICK*
And Red agree to trade: *yes*
Status change: *status*

```
Enter a command:
> trade
> Red
> WIFI
> BRICK
> > > Orange offers Red one WIFI for one BRICK.
Does Red accept this offer?
> yes
Orange gains one BRICK and loses one WIFI,
Red gains one WIFI and loses one BRICK.
Enter a command:
> status
Builder Blue    has 2 building points, 1 BRICK, 2 ENERGY, 3 GLASS, 4 HEAT, 5 WIFI.
Builder Red     has 4 building points, 0 BRICK, 1 ENERGY, 1 GLASS, 0 HEAT, 1 WIFI.
Builder Orange  has 9 building points, 101 BRICK, 100 ENERGY, 100 GLASS, 100 HEAT, 99 WIFI.
Builder Yellow  has 2 building points, 2 BRICK, 2 ENERGY, 2 GLASS, 2 HEAT, 3 WIFI.
Enter a command:
>
```

Enter *next roll 2 status*

```
Enter a command:
> status
Builder Blue    has 2 building points, 1 BRICK, 2 ENERGY, 3 GLASS, 4 HEAT, 5 WIFI.
Builder Red     has 4 building points, 0 BRICK, 1 ENERGY, 1 GLASS, 0 HEAT, 1 WIFI.
Builder Orange  has 9 building points, 101 BRICK, 100 ENERGY, 100 GLASS, 100 HEAT, 99 WIFI.
Builder Yellow  has 2 building points, 2 BRICK, 2 ENERGY, 2 GLASS, 2 HEAT, 3 WIFI.
Enter a command:
>
```

Now let's test market
If you don't have enough resource (<4) but you try to market: *market WIFI HEAT status*
Market failed, and status remains same:

```
Enter a command:
> market
> WIFI
> HEAT
You don't have enough WIFI. You need 4.Enter a command:
> status
Builder Blue    has 2 building points, 1 BRICK, 2 ENERGY, 3 GLASS, 4 HEAT, 5 WIFI.
Builder Red     has 4 building points, 0 BRICK, 1 ENERGY, 1 GLASS, 0 HEAT, 1 WIFI.
Builder Orange  has 9 building points, 101 BRICK, 100 ENERGY, 100 GLASS, 100 HEAT, 99 WIFI.
Builder Yellow  has 2 building points, 2 BRICK, 2 ENERGY, 2 GLASS, 2 HEAT, 3 WIFI.
Enter a command:
>
```

Enter next roll 2

When successfully market: *market HEAT WIFI status*

```
Enter a command:
> market
> HEAT
> WIFI
Blue gains one WIFI and loses four HEAT.
Enter a command:
> status
Builder Blue    has 2 building points, 1 BRICK, 2 ENERGY, 3 GLASS, 0 HEAT, 6 WIFI.
Builder Red     has 4 building points, 0 BRICK, 1 ENERGY, 1 GLASS, 0 HEAT, 1 WIFI.
Builder Orange  has 9 building points, 101 BRICK, 100 ENERGY, 100 GLASS, 100 HEAT, 99 WIFI.
Builder Yellow  has 2 building points, 2 BRICK, 2 ENERGY, 2 GLASS, 2 HEAT, 3 WIFI.
Enter a command:
>
```

Also, you may not market the same resource: *market WIFI WIFI*

```
Enter a command:
> market
> wifi
> wifi
> Why are you buying the same resource?
Enter a command:
>
```

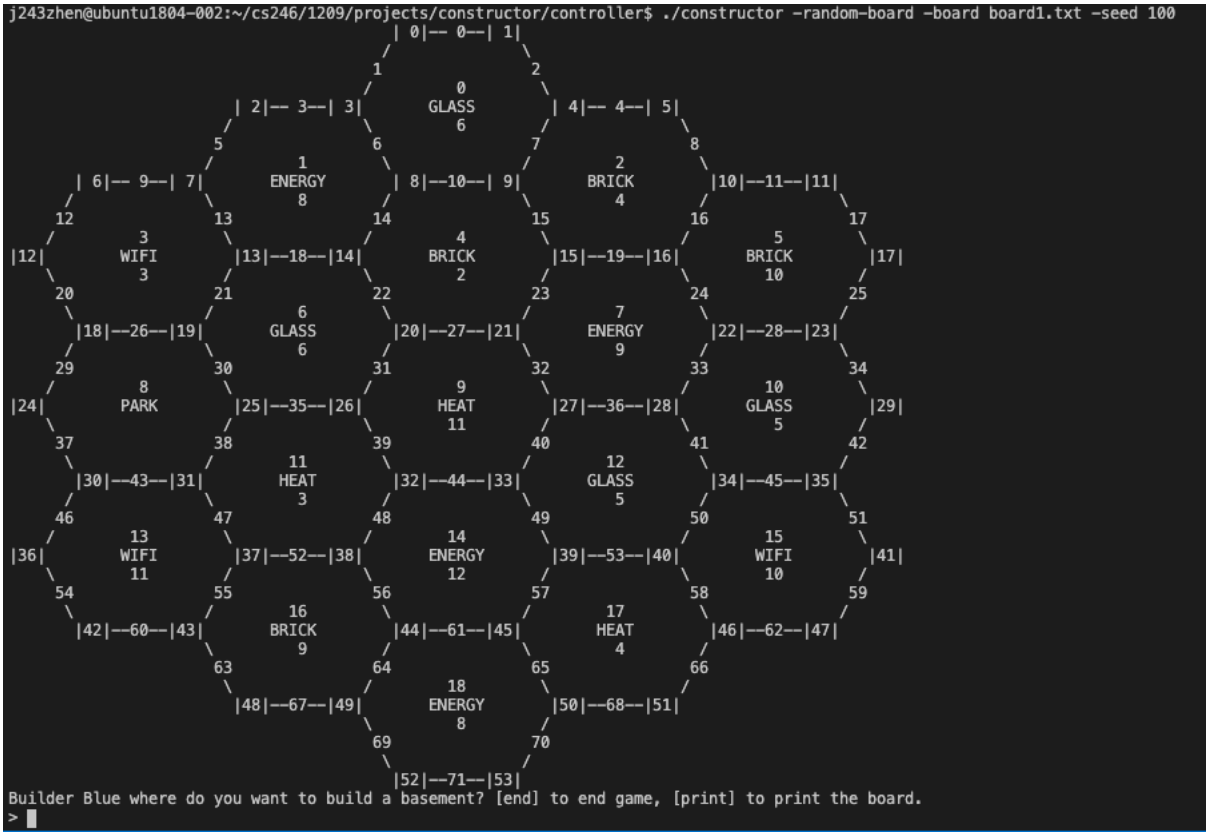
Enter *end*

Invalid or special commands when initializing a board:

There are some invalid or special commands involved in the project, especially regarding the command line arguments.

As required by the instruction, if there is a loaded board/game, the -random-board command will be ignored.

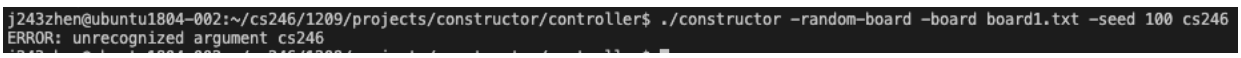
Type ./constructor -random-board -board1.txt -seed 120



The board is the actual board in board1.txt, not a random one, not associate with the seed number 120. The seed number here only controls the number of fair dice.

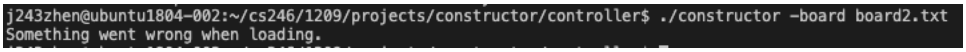
Note that -load and -board cannot be used together.

As long as invalid command is entered, the game will not be started.

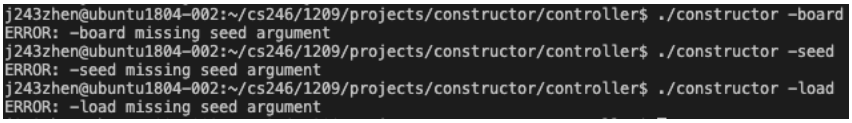


Loaded board/game can also have new seed to control the number of dice, for example, we type ./constructor -board.txt -seed 9, the result of first fair dice is still 2.

The argument -board can also detect invalid board. In board2.txt, one dice number is 13, and it is detected by the program.



If any command line argument is missing, the game will not start.



All supported files in our testing are submitted with my project on Marmoset.