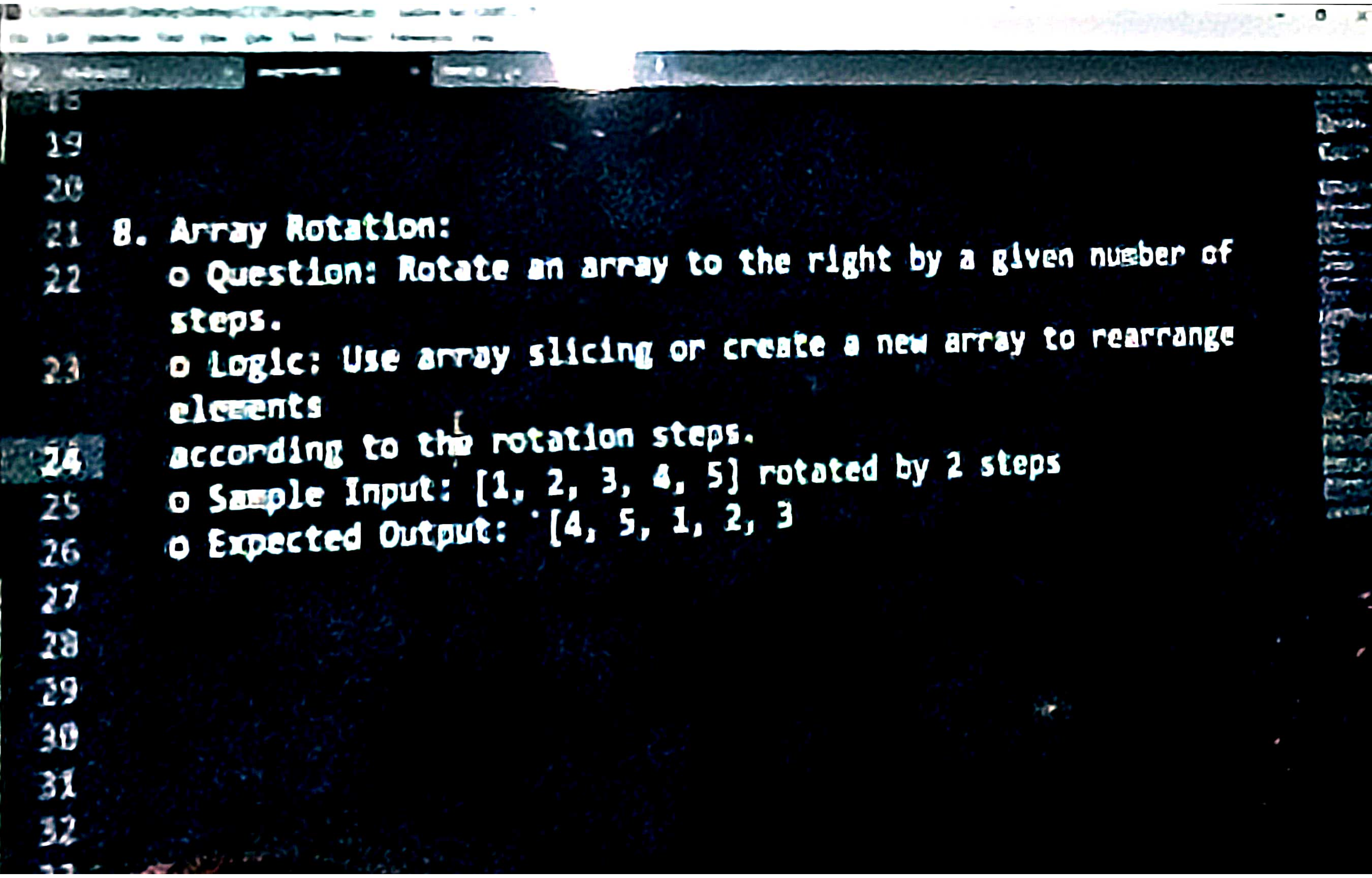


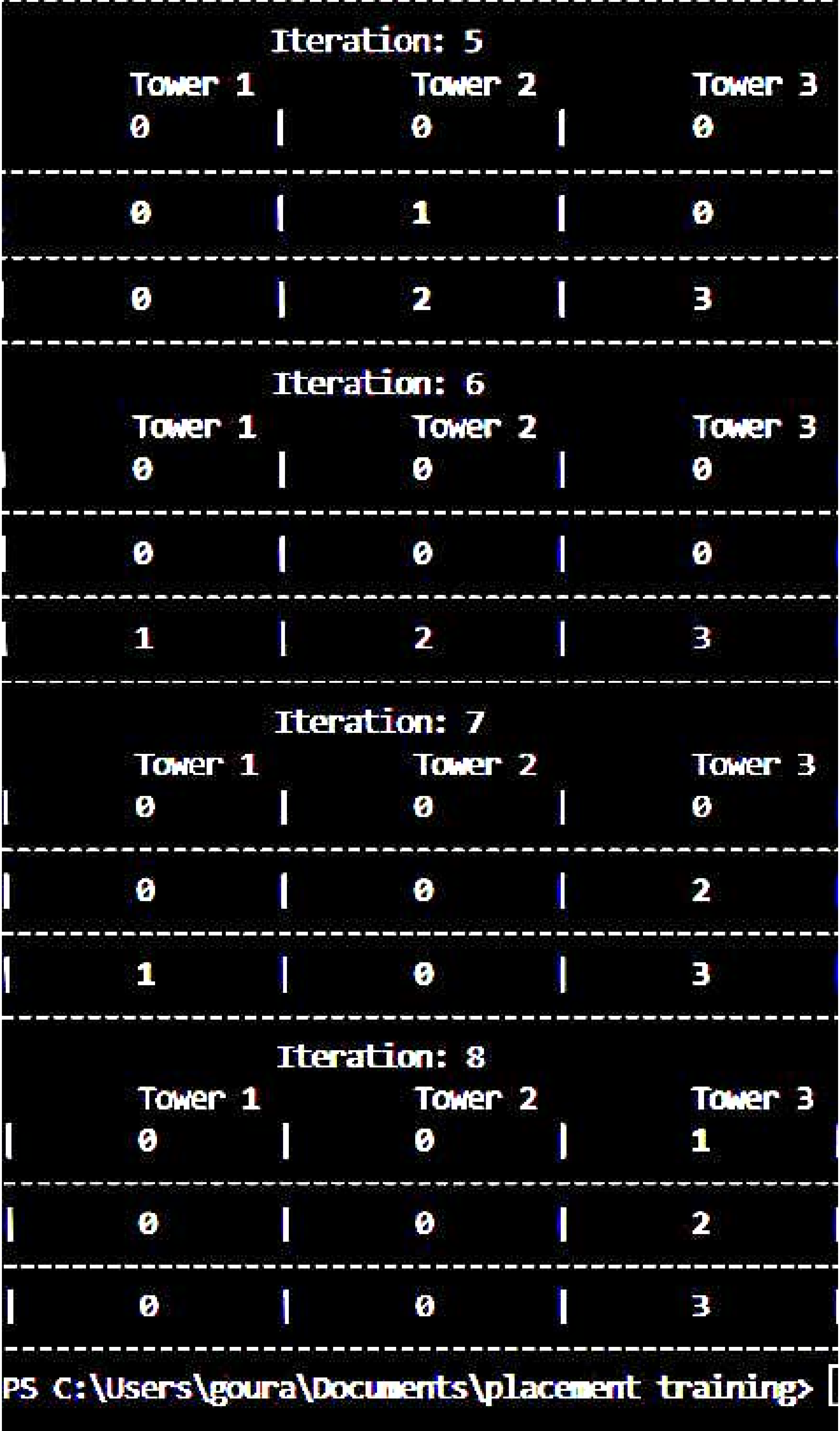
8. Array Rotation:

o Question: Rotate an array to the right by a given number of steps.

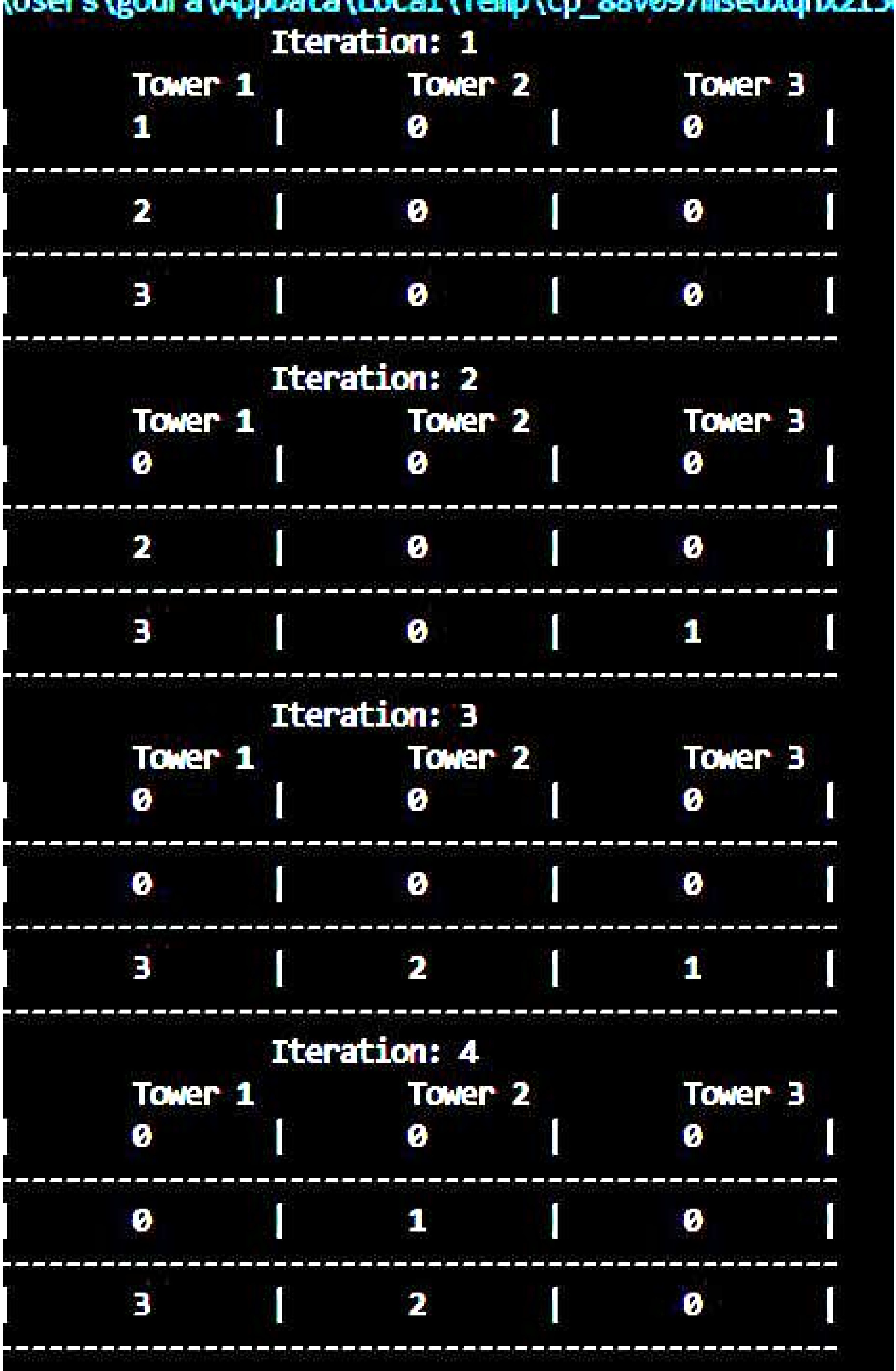
o Logic: Use array slicing or create a new array to rearrange elements
according to the rotation steps.

o Sample Input: [1, 2, 3, 4, 5] rotated by 2 steps

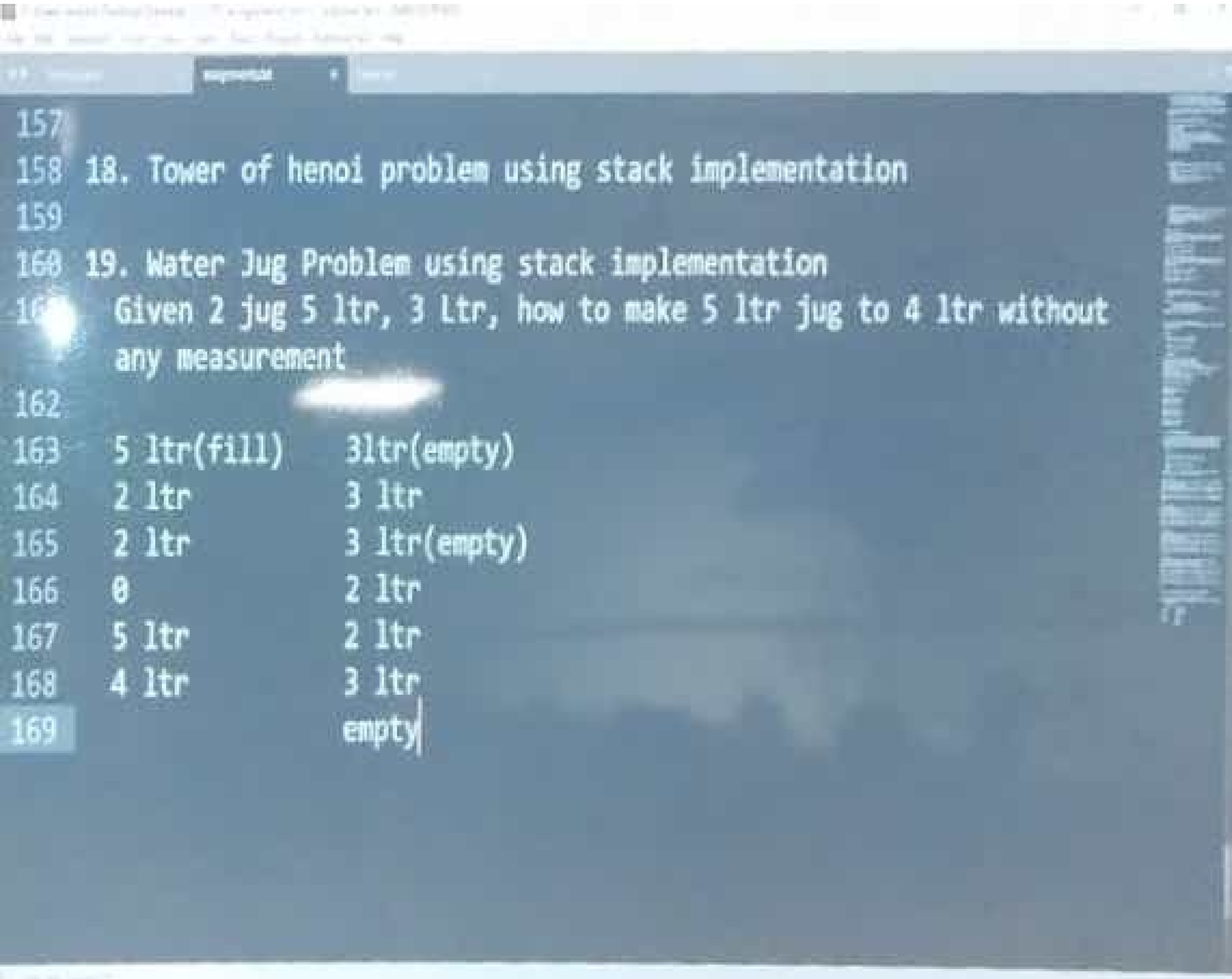
o Expected Output: [4, 5, 1, 2, 3



```
------------------------------------------------------------
                   Iteration: 5
         Tower 1            Tower 2            Tower 3
         0         |        0         |        0
------------------------------------------------------------
         0         |        1         |        0
------------------------------------------------------------
         0         |        2         |        3
------------------------------------------------------------
                   Iteration: 6
         Tower 1            Tower 2            Tower 3
         0         |        0         |        0
------------------------------------------------------------
         0         |        0         |        0
------------------------------------------------------------
         1         |        2         |        3
------------------------------------------------------------
                   Iteration: 7
         Tower 1            Tower 2            Tower 3
|        0         |        0         |        0
------------------------------------------------------------
|        0         |        0         |        2
------------------------------------------------------------
|        1         |        0         |        3
------------------------------------------------------------
                   Iteration: 8
         Tower 1            Tower 2            Tower 3
|        0         |        0         |        1
------------------------------------------------------------
|        0         |        0         |        2
------------------------------------------------------------
|        0         |        0         |        3
------------------------------------------------------------
PS C:\Users\goura\Documents\placement training>
```



```
\Users\gourav\AppData\Local\Temp\cp_88v097mseuxqnx213
                    Iteration: 1
        Tower 1             Tower 2             Tower 3
|       1         |         0         |         0         |
-------------------------------------------------------------
|       2         |         0         |         0         |
-------------------------------------------------------------
|       3         |         0         |         0         |
-------------------------------------------------------------
                    Iteration: 2
        Tower 1             Tower 2             Tower 3
|       0         |         0         |         0         |
-------------------------------------------------------------
|       2         |         0         |         0         |
-------------------------------------------------------------
|       3         |         0         |         1         |
-------------------------------------------------------------
                    Iteration: 3
        Tower 1             Tower 2             Tower 3
|       0         |         0         |         0         |
-------------------------------------------------------------
|       0         |         0         |         0         |
-------------------------------------------------------------
|       3         |         2         |         1         |
-------------------------------------------------------------
                    Iteration: 4
        Tower 1             Tower 2             Tower 3
|       0         |         0         |         0         |
-------------------------------------------------------------
|       0         |         1         |         0         |
-------------------------------------------------------------
|       3         |         2         |         0         |
-------------------------------------------------------------
```

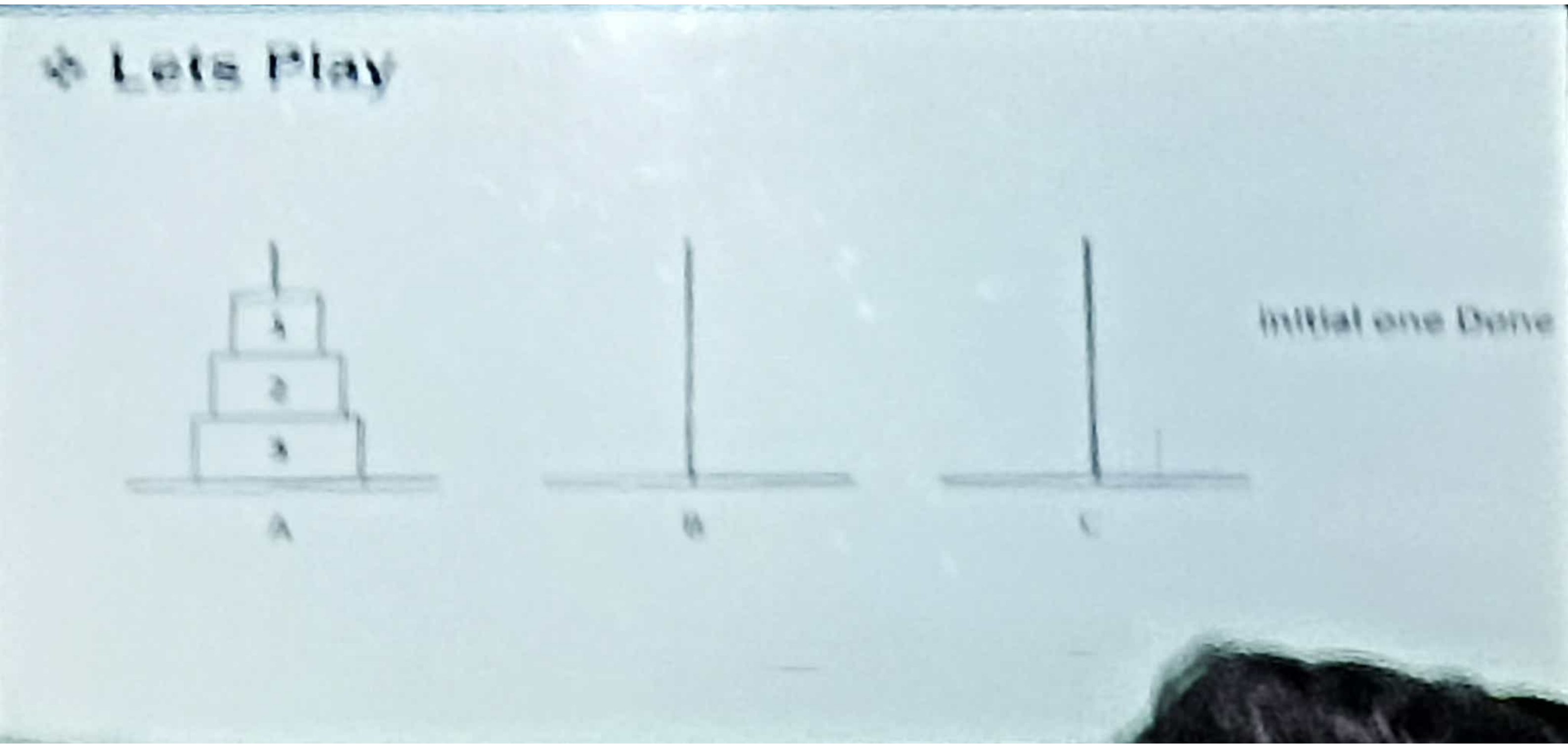


18. Tower of henoi problem using stack implementation

19. Water Jug Problem using stack implementation

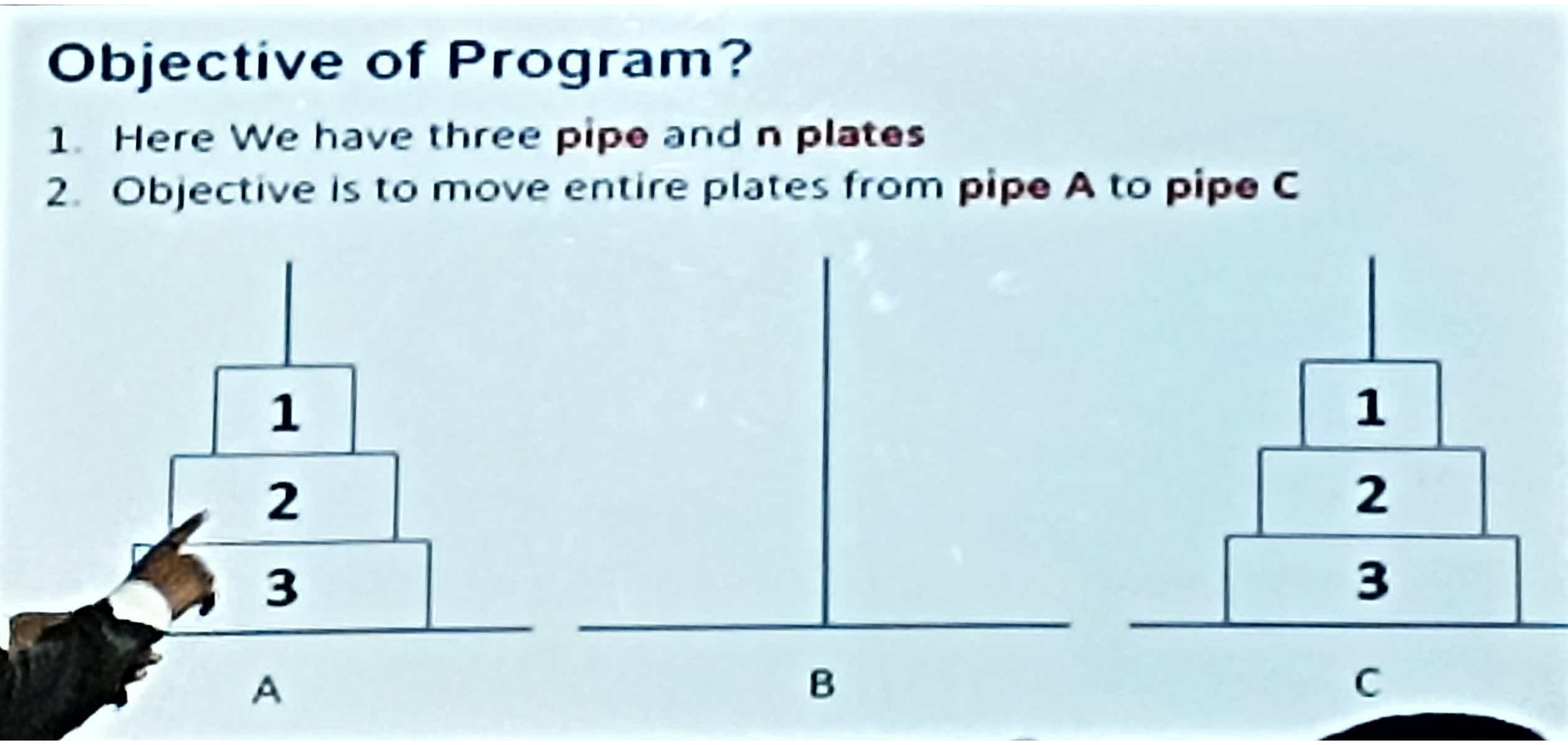
Given 2 jug 5 ltr, 3 Ltr, how to make 5 ltr jug to 4 ltr without any measurement

5 ltr(fill)	3ltr(empty)
2 ltr	3 ltr
2 ltr	3 ltr(empty)
0	2 ltr
5 ltr	2 ltr
4 ltr	3 ltr
	empty



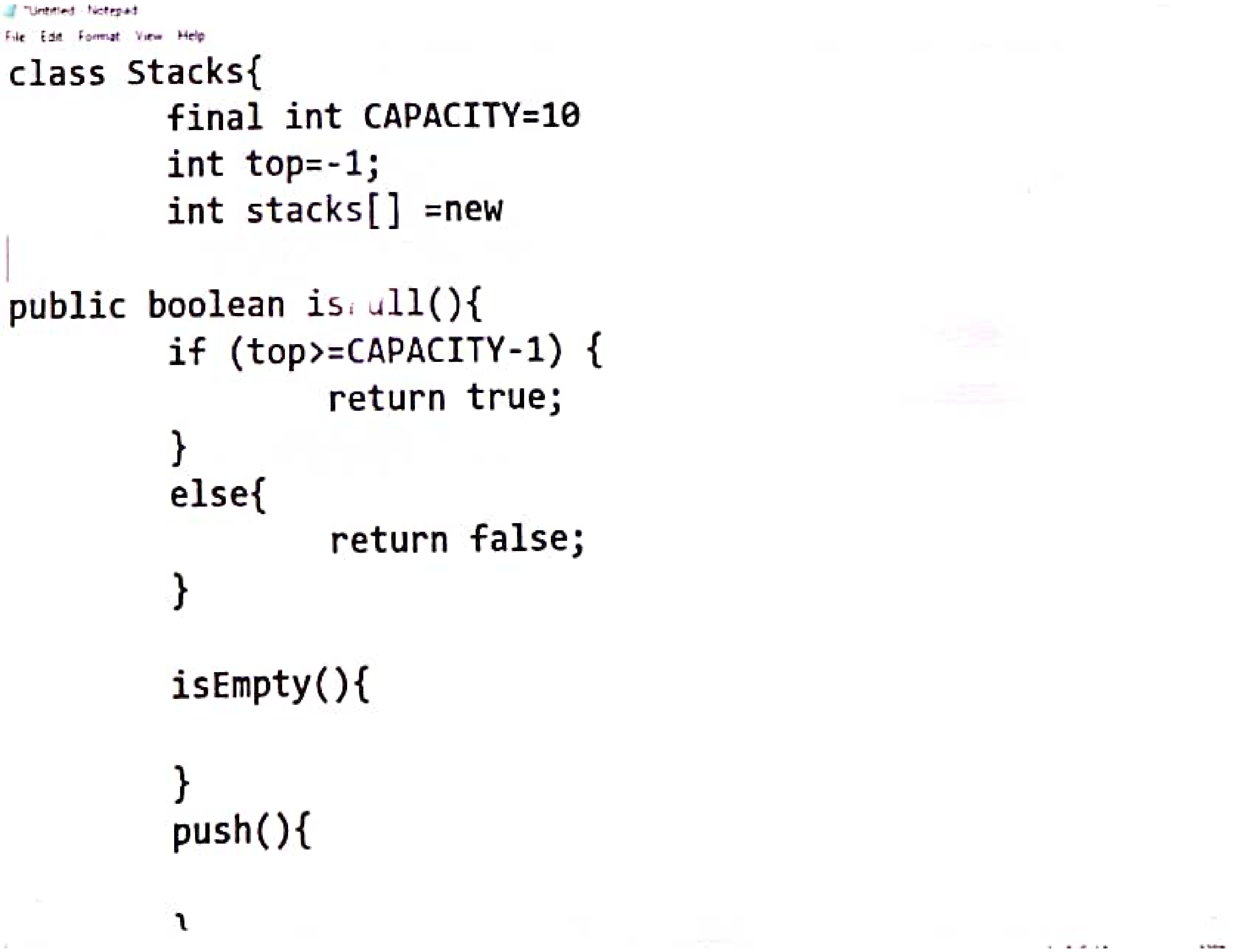
❖ Lets Play

Initial one Done

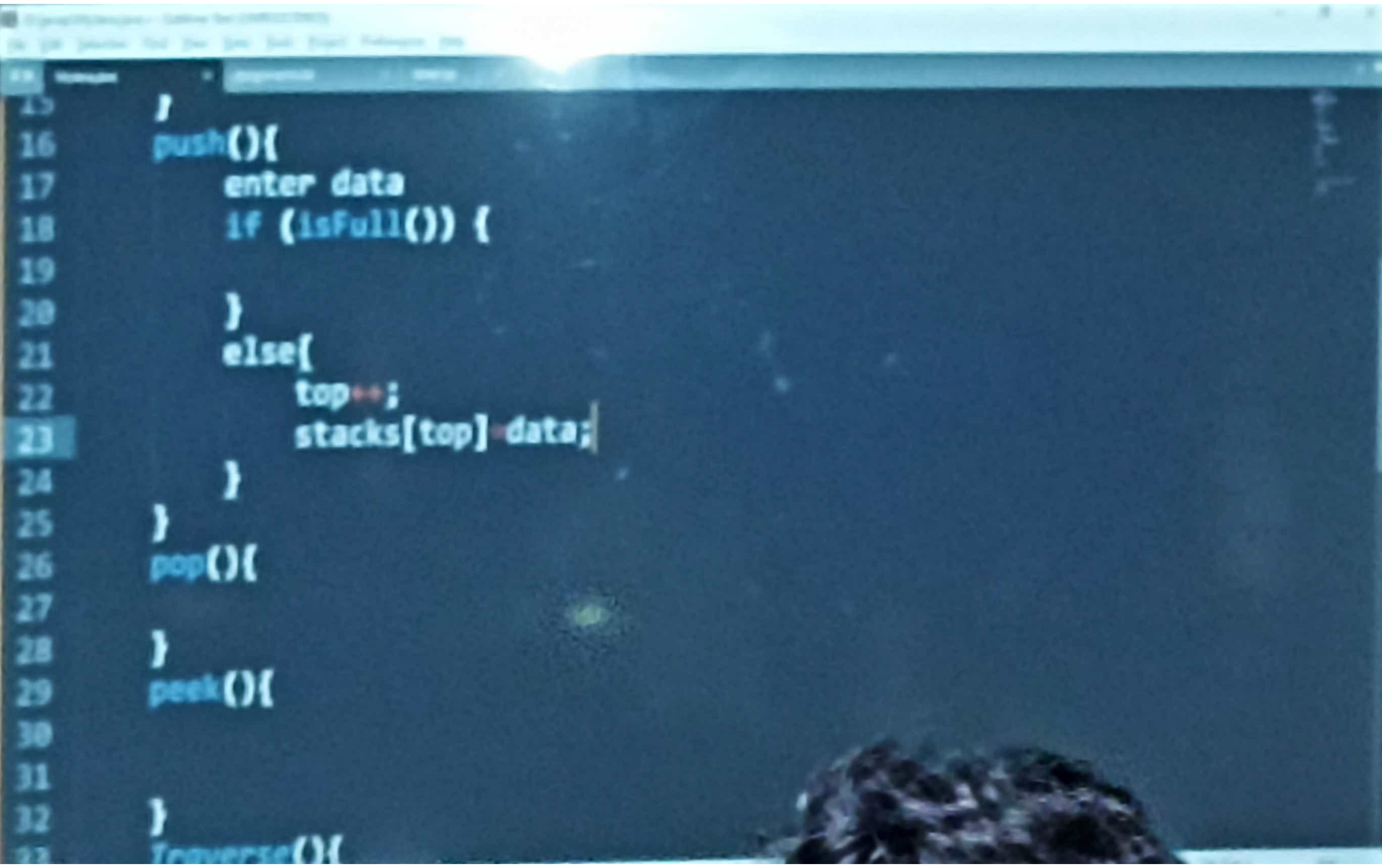


Objective of Program?

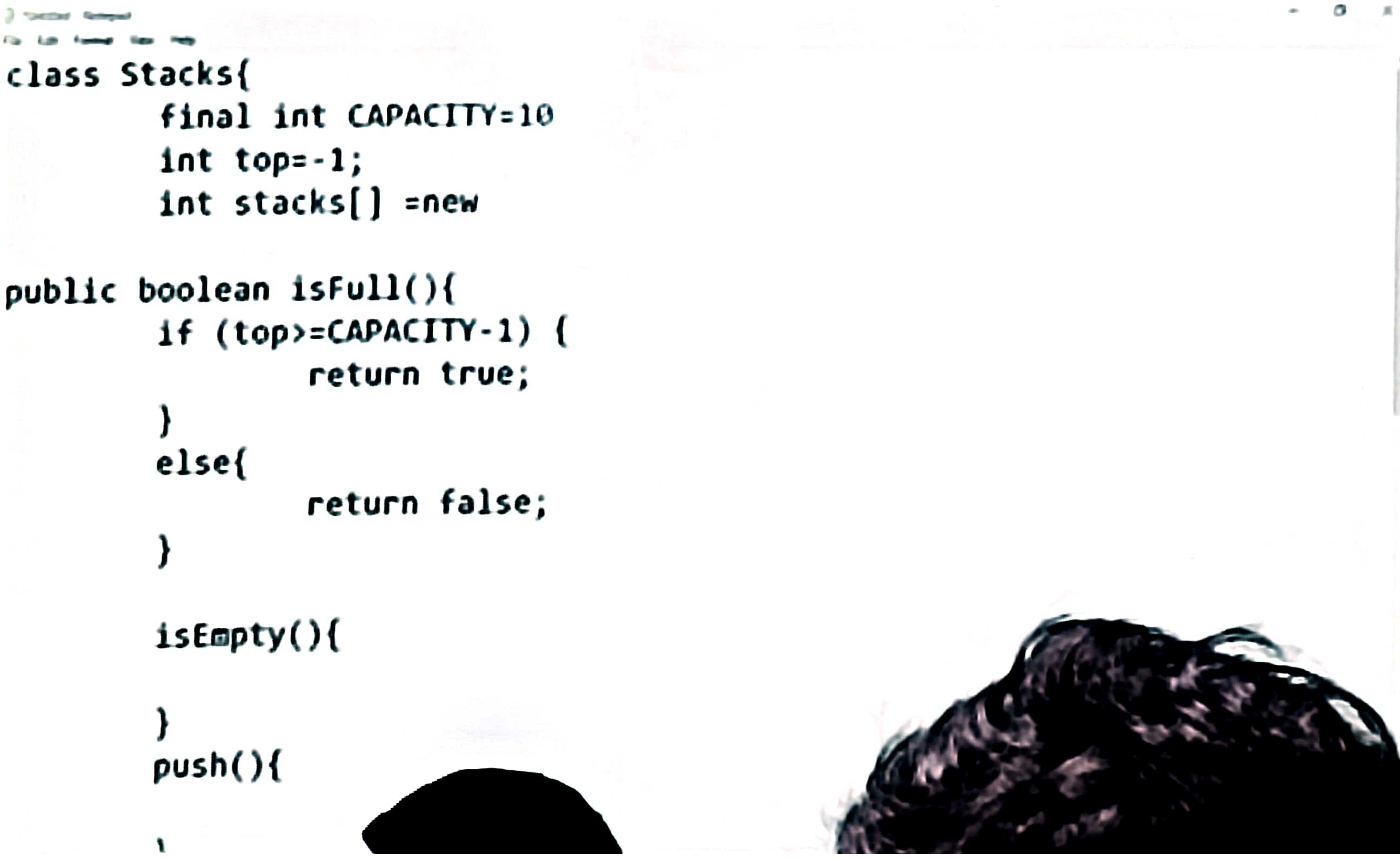
1. Here We have three **pipe** and **n plates**
2. Objective is to move entire plates from **pipe A** to **pipe C**



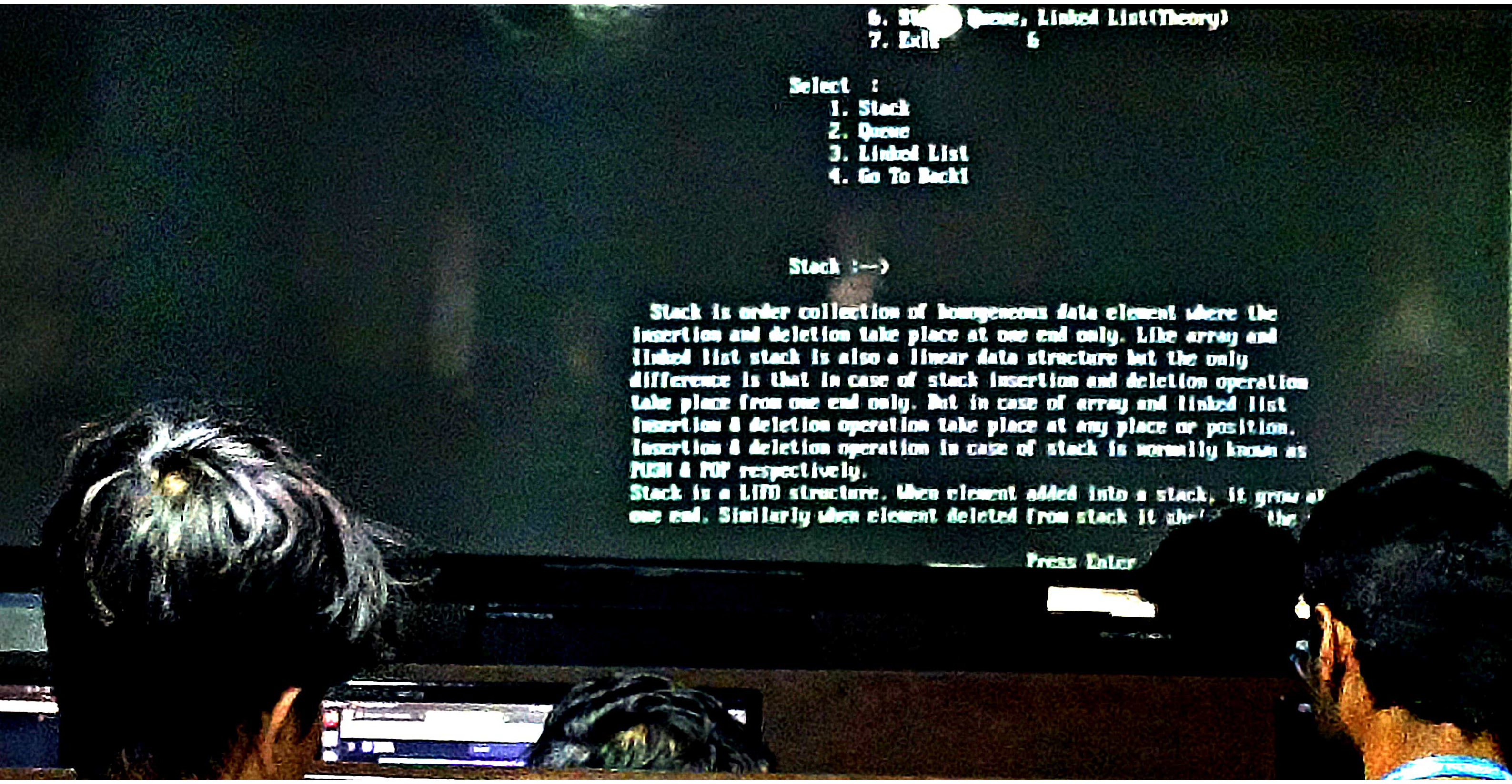
```
class Stacks{
        final int CAPACITY=10
        int top=-1;
        int stacks[] =new

public boolean isFull(){
        if (top>=CAPACITY-1) {
                return true;
        }
        else{
                return false;
        }

        isEmpty(){

        }
        push(){

        }
```



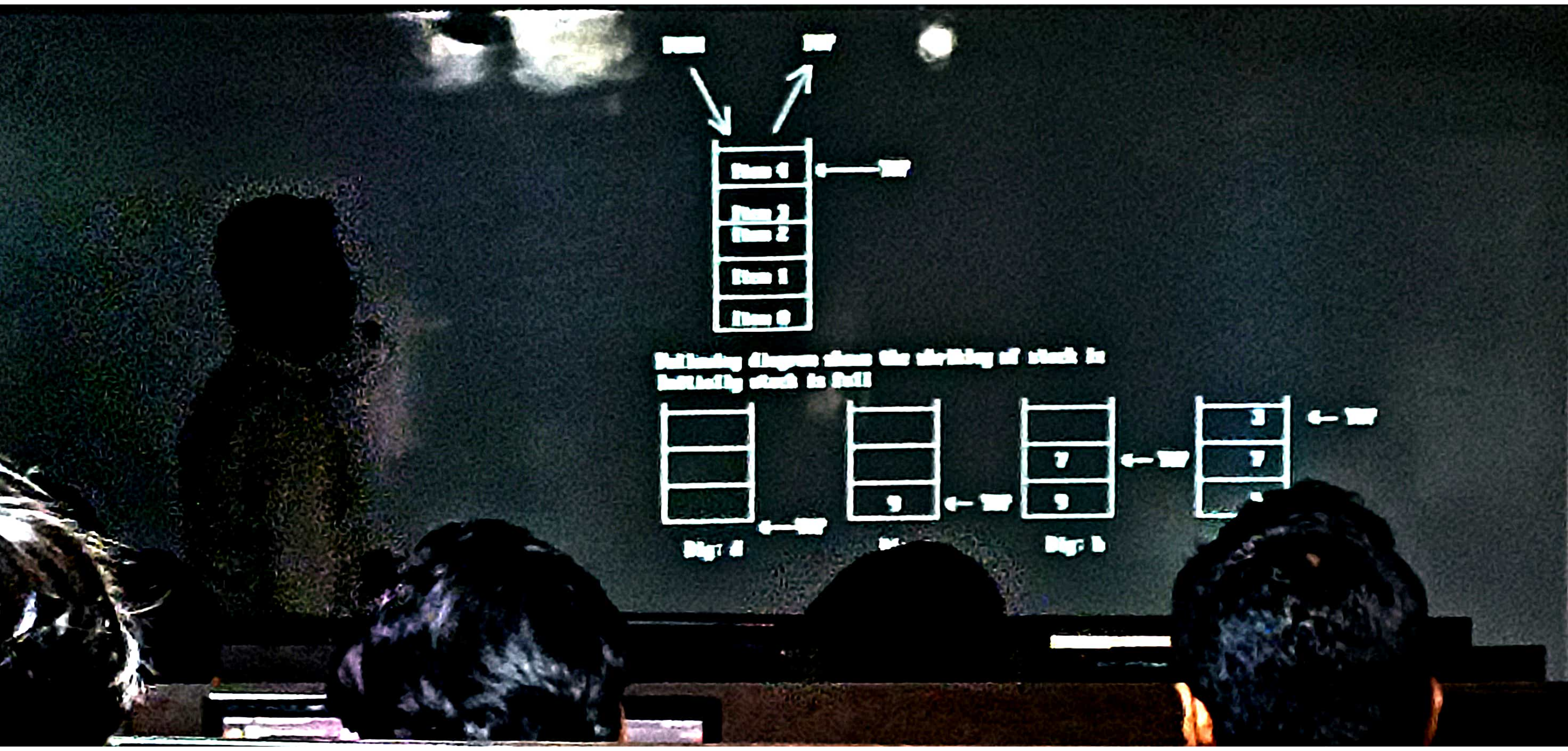
```
  }
  push(){
      enter data
      if (isFull()) {

      }
      else{
          top++;
          stacks[top]=data;
      }
  }
  pop(){

  }
  peek(){


  }
  Traverse(){
```



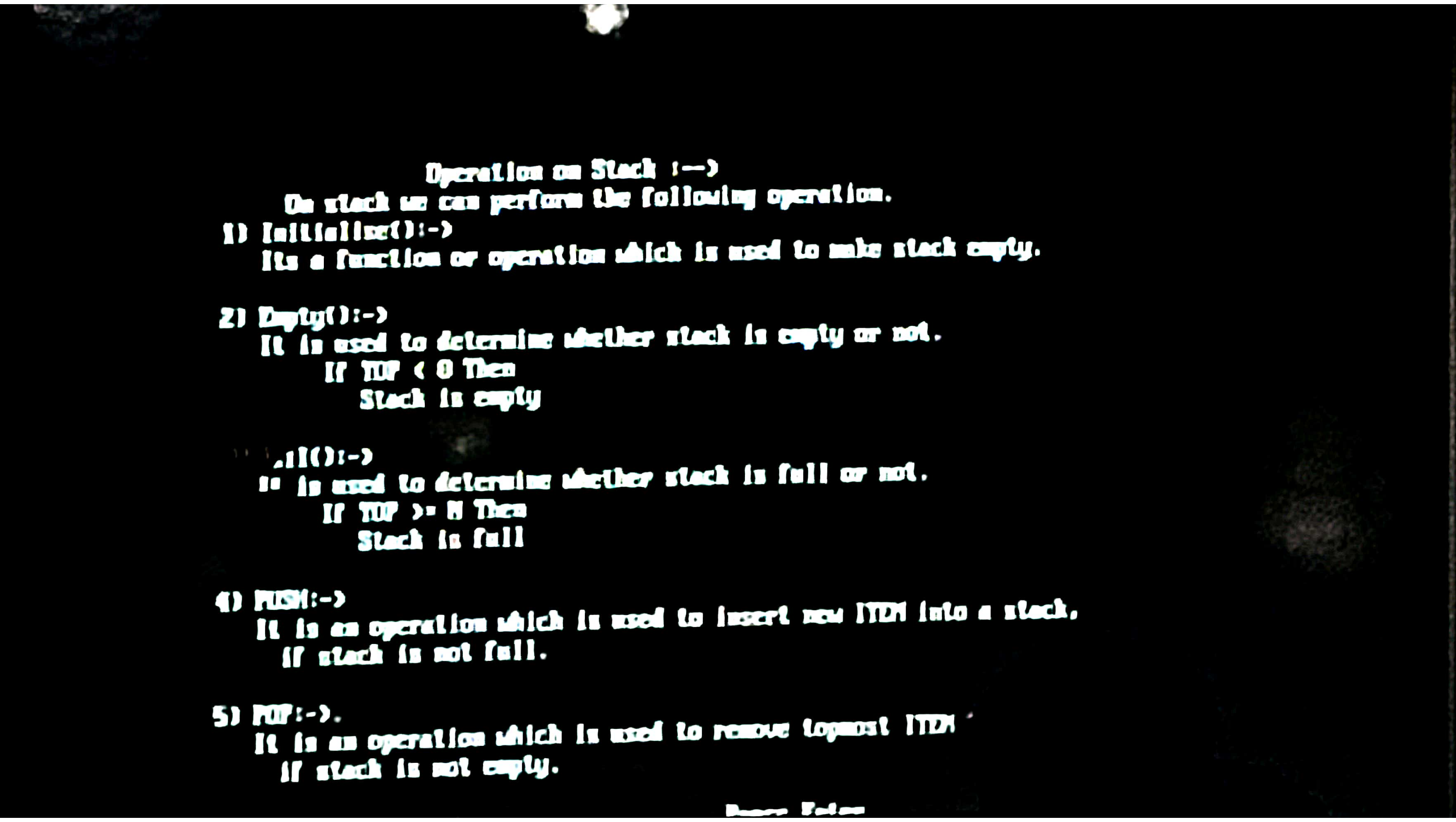
```
class Stacks{
        final int CAPACITY=10
        int top=-1;
        int stacks[] =new

public boolean isFull(){
        if (top>=CAPACITY-1) {
                return true;
        }
        else{
                return false;
        }

        isEmpty(){

        }
        push(){

        }
```



```
                    6. Stack, Queue, Linked List(Theory)
                    7. Exit        6

               Select  :
                    1. Stack
                    2. Queue
                    3. Linked List
                    4. Go To Back!


               Stack :-->

  Stack is order collection of homogeneous data element where the
insertion and deletion take place at one end only. Like array and
linked list stack is also a linear data structure but the only
difference is that in case of stack insertion and deletion operation
take place from one end only. But in case of array and linked list
insertion & deletion operation take place at any place or position.
Insertion & deletion operation in case of stack is normally known as
PUSH & POP respectively.
Stack is a LIFO structure. When element added into a stack, it grow at
one end. Similarly when element deleted from stack it sh[illegible] the

                                        Press Enter
```



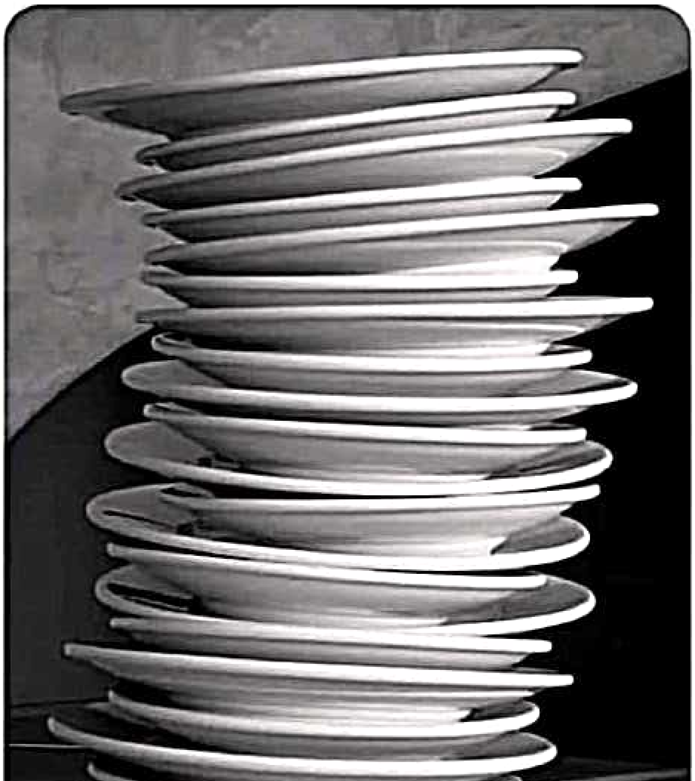
Following diagram shows the working of stack in building stack is full



```
                    Operation on Stack :->
              On stack we can perform the following operation.
   1) Initialize():->
      Its a function or operation which is used to make stack empty.

   2) Empty():->
      It is used to determine whether stack is empty or not.
           If TOP < 0 Then
              Stack is empty

   3) Full():->
      It is used to determine whether stack is full or not.
           If TOP >= N Then
              Stack is full

   4) PUSH:->
      It is an operation which is used to insert new ITEM into a stack,
        if stack is not full.

   5) POP:->.
      It is an operation which is used to remove topmost ITEM
        if stack is not empty.

                                        Press Enter
```

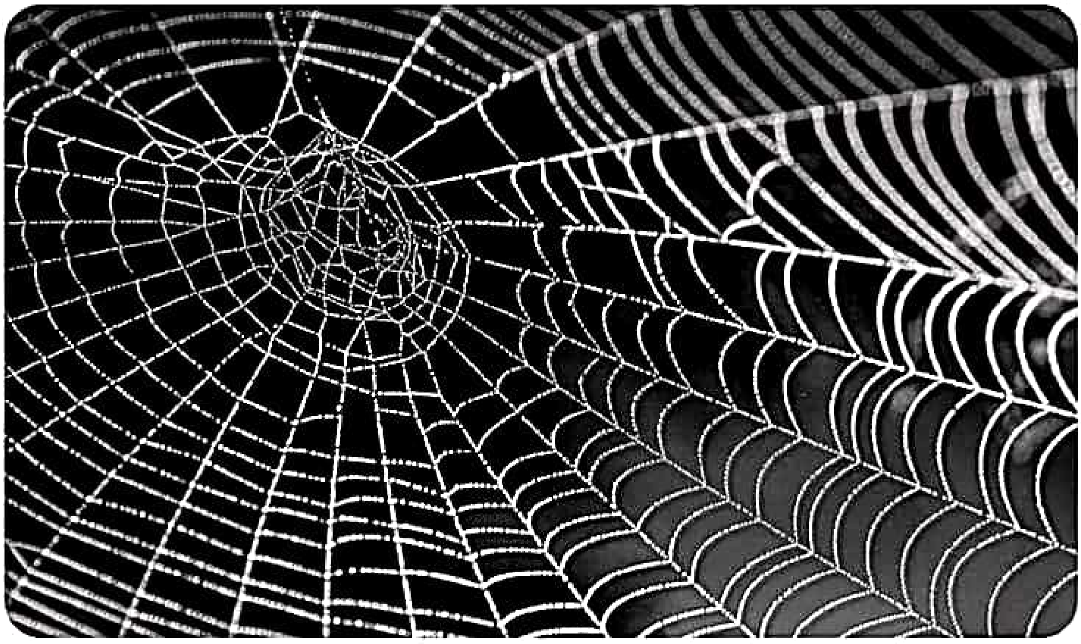
Stack

new plates are added from the top
and also removed from the top



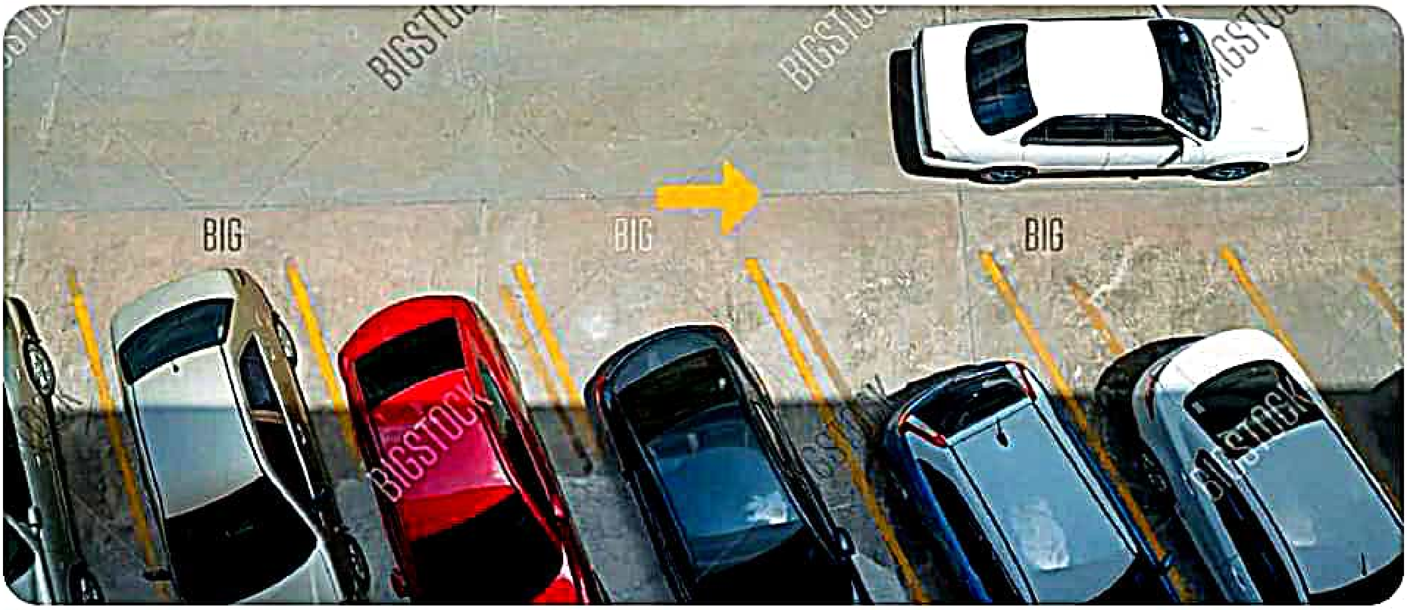
Graph

a spider web



Array

cars parked in a single line

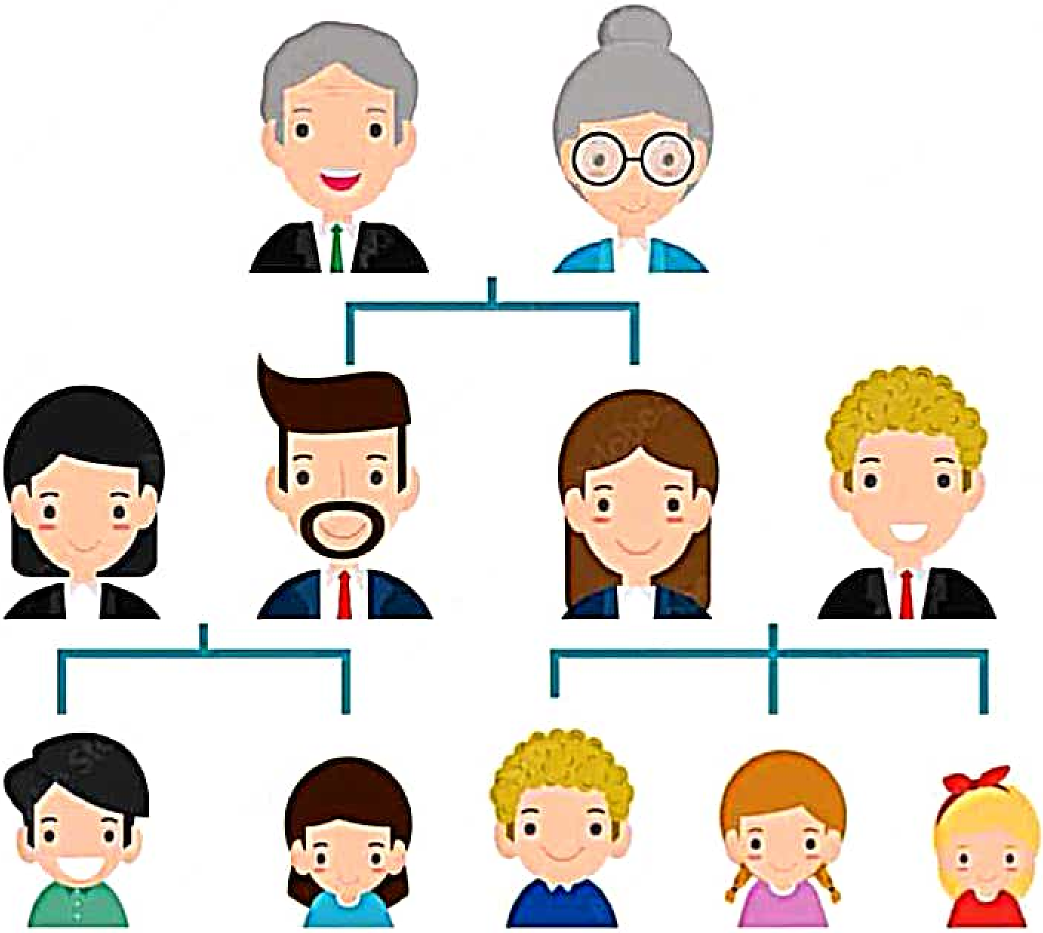


Linked List

a train

Tree

a family tree

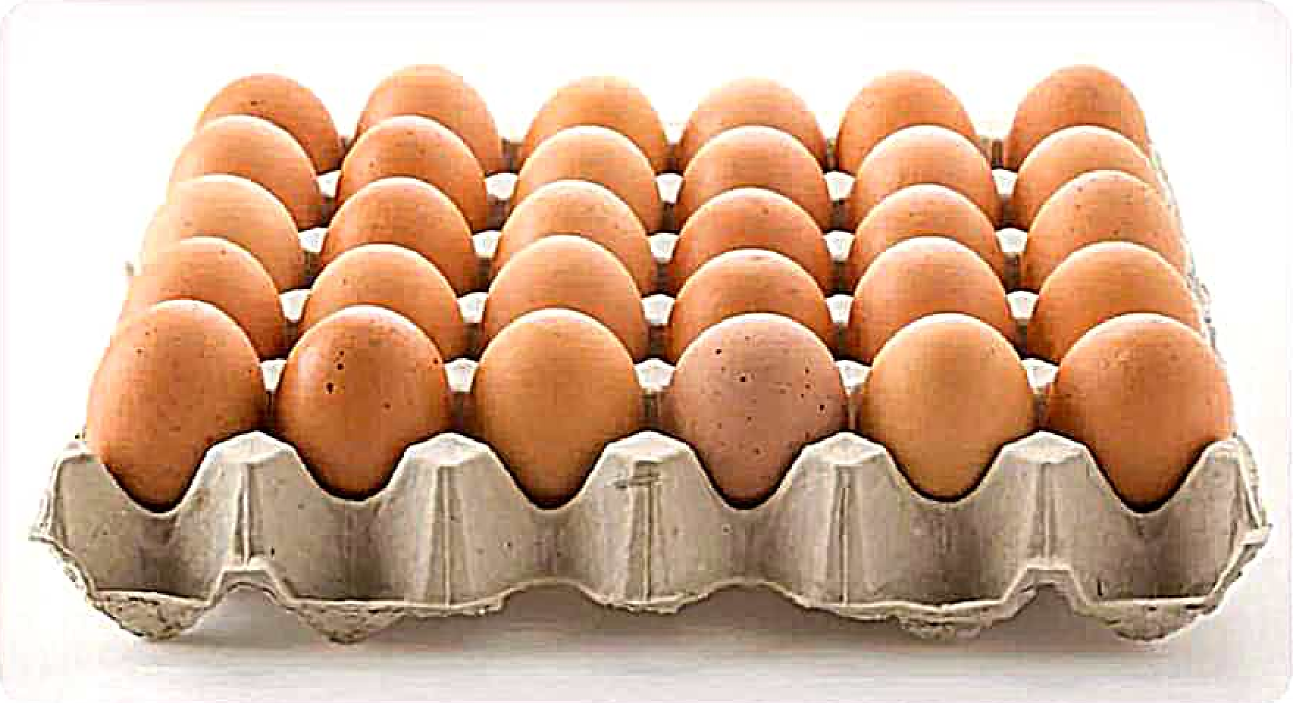


Queue

waiting in line to buy coffee

2D - Array

tray of eggs



3D - Array

rubiks cube