Linked list Basics

Music: "Plenty times" by Frank and Walters

Agenda:

1. ~~Classes and object~~ Basics
   a. ~~constructor~~ LLD
2. linked list introduction
   a. ~~implementation using class~~
   b. pros and cons
3. Problems
   a. Size of the linked list
   b. Print the linked list
   c. insertion

Problem: Before object oriented programming.

↳ store students name and marks, adres

- int
- string
- char
- array
- hashmap

eg.

Vitul	85%	Delhi
Manoj	80%	Mumbai
Sumit	90%	Bhopal
String	int	string

⇒ use our custom defined data type

⇒ class Student {

```
string  student_name;
int     marks;
string  adress;
```

}

class
struct

* washing machine:

① some properties (color, brand, size, price..)
② some functionalities (washing, drying ...)

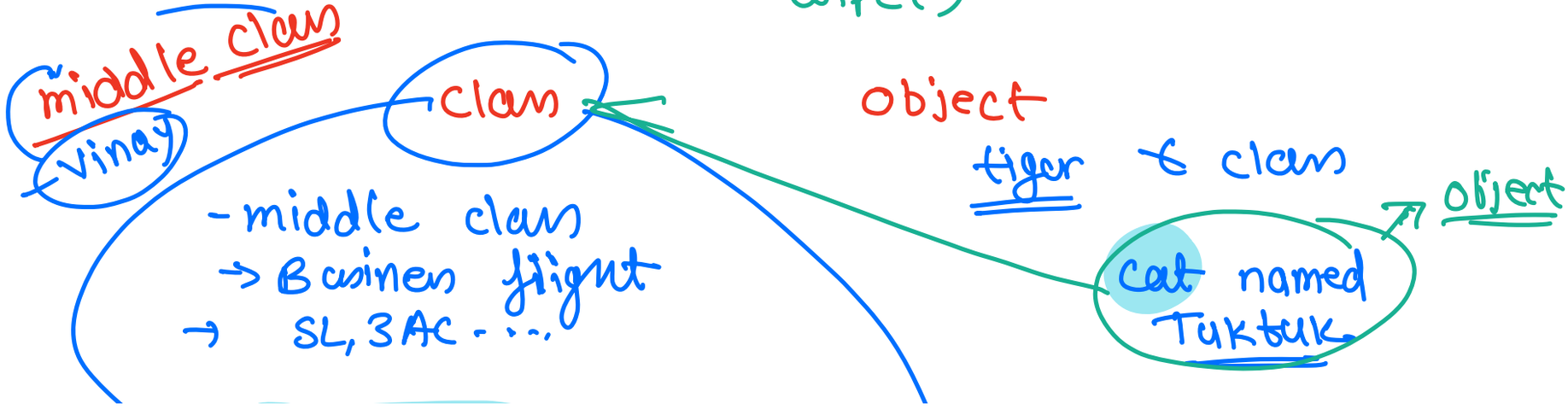
⇒ model the real world in programming via object oriented programming

⇒ every object (programming) has some properties and some functionalities.

(*) car :

color
seats
brand

accelarate()
break()
play-music()
wipe()



Categorization of real world objects
Animals
employee
student

* class ⇒ template / Blueprint for defining the objects

* Object ⇒ is an instance of the class
  (real world example)

```
class Student {
    char[12]  student_name;
    int       roll number:
    int       marks;
    char[12]  address;
}
```

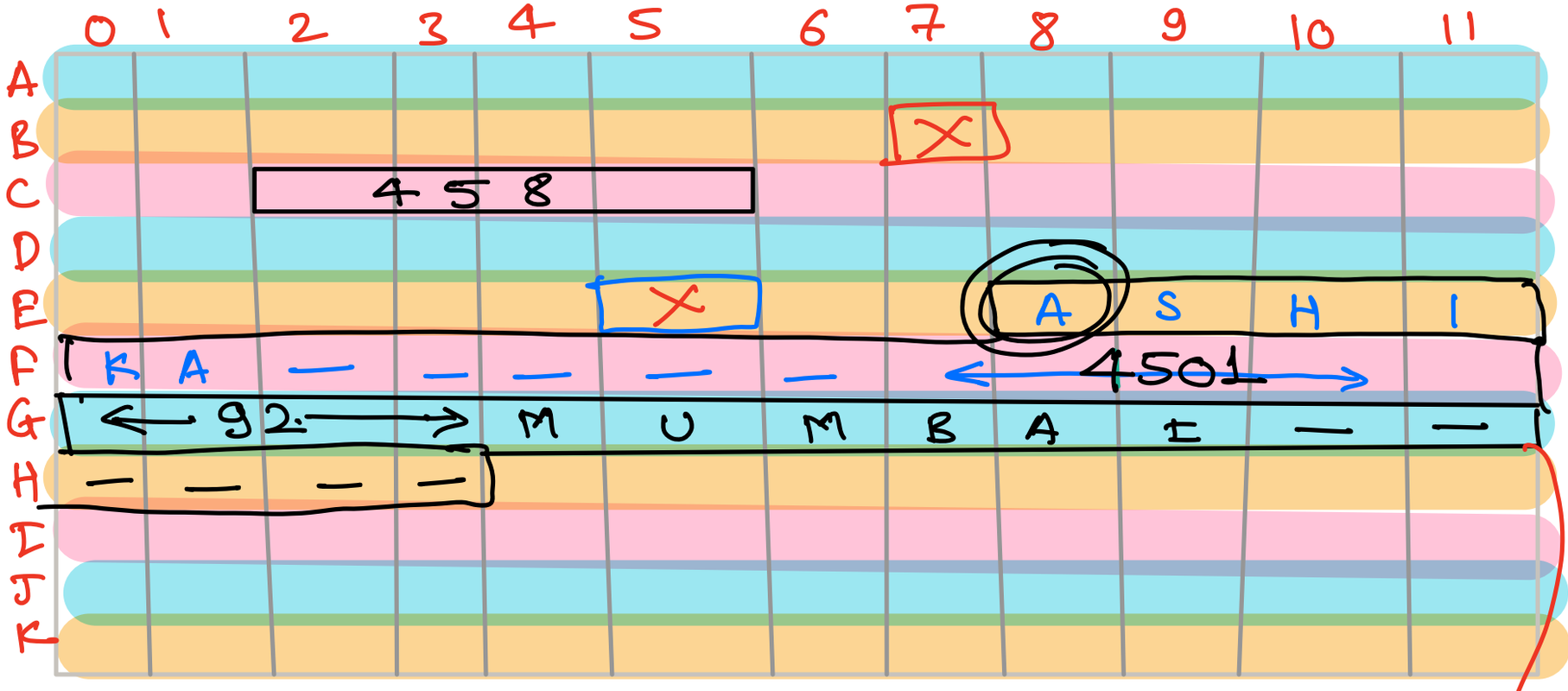
12 Bytes ⇐ char[12] student_name;
4 Bytes ⇐ int roll number:
4 Bytes ⇐ int marks;
12 Bytes ⇐ char[12] address;

name of the class → start with capital letter

* Student [Ashika, 5101, 92, Mumbai)

⇒ Student A = new Student();

declare an object



each block represent 1 Byte 8 bits

int → 4 byte

a = 458 → C2

4 Byte

which memory adress we need to read the data

for the student "Ashika".

```
Student A = new Student();
```

e8 — memory location

```
A.student_name = "Ashika"
A.rollnumber   = 5101
A.marks        = 92
A.addren       = "Mumbai"
```

(e8, 12 Bytes)
(f7, 4 Bytes)

```
class Student {
    char[12] student_name;
    int roll_number;
    int m1;
    int m2;
    int m3;

    int totalMarks() {
        return m1 + m2 + m3;
    }
```

properties

functionality — total marks of a student.

functionality

```
}
Student(char[] name, int roll_number
                        int m1, int m2, int m3) {
this.Student_name = name
 this.roll_number = roll_number;                    ;
  this.  m1 = m1
  this.  m2 = m2
   this. m3 = m3
}
}
```

S1.Stude = "Vitul"

1 → Student S1 = new Student();

2 → Student S1 = new Student("vitul", 102, 90, 91, 92)

Constructor has the same name as the class name.

⇒ S1.totalmarks() = 273

→ denotes the addres where the object is stored.

S2 = null

⇒ Student S2;
output of S2.totalmarks();

null pointer exceptio)

Error: object is not defined

---

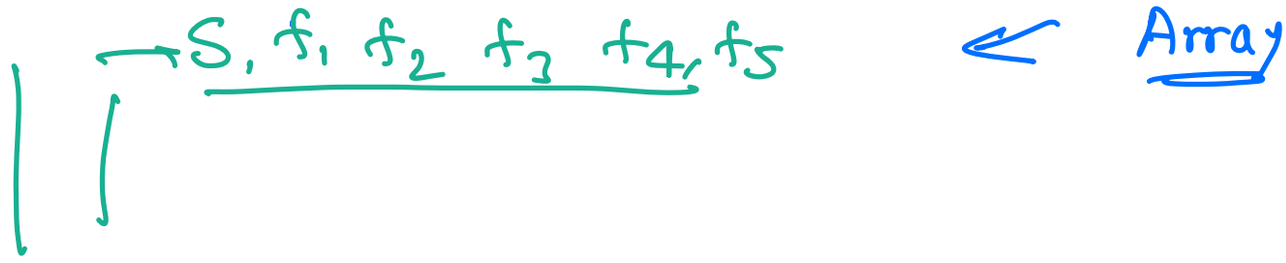
§. Student S3 = S1 // same object as S1 would be referred

Output S3.totalmarks()? = 273

---

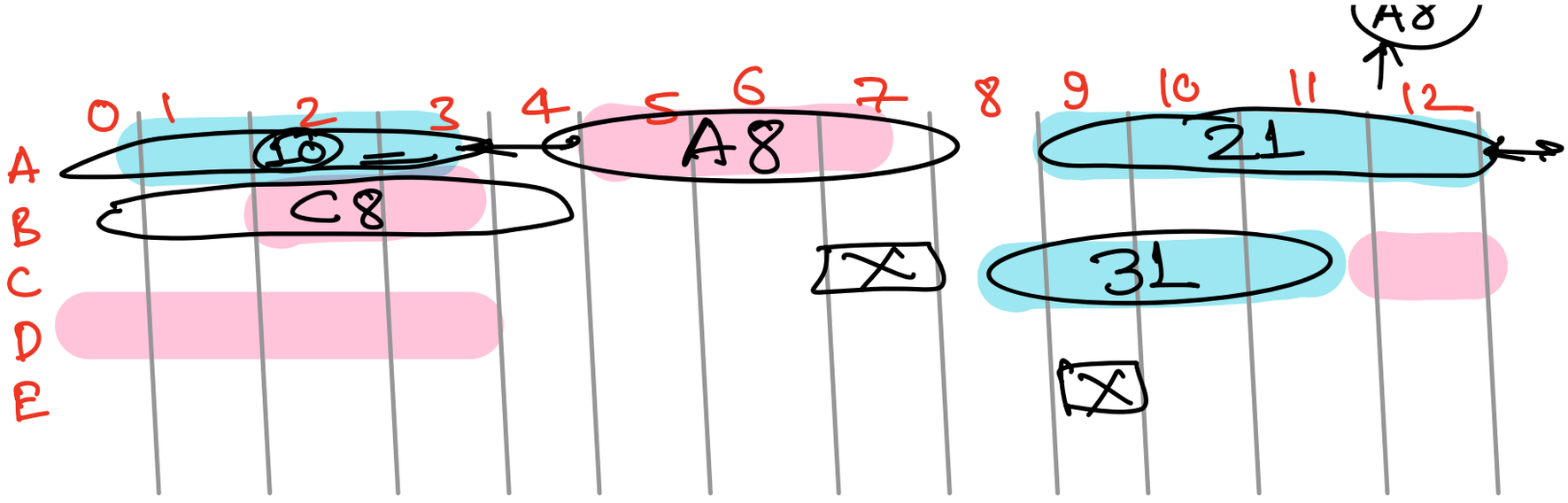
§. Shubham's b'day : 5 friends
↳ movie, popcorns

⇒ they would like to sit together.



§. they don't get consecutive seats

Just remember the next person's location.

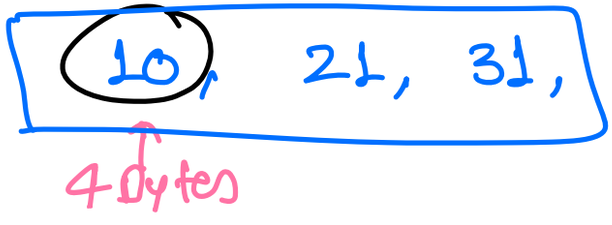


Store 6 int ⇒ 6*4 ⇒ 24 Bytes

↳ Array requires 24 Bytes

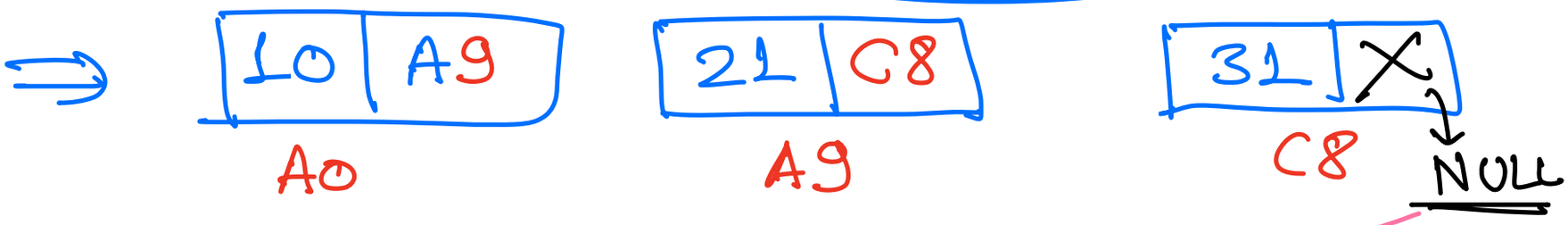
not possible to store 6 int in an array in the given memory.

⇒ toy linked list



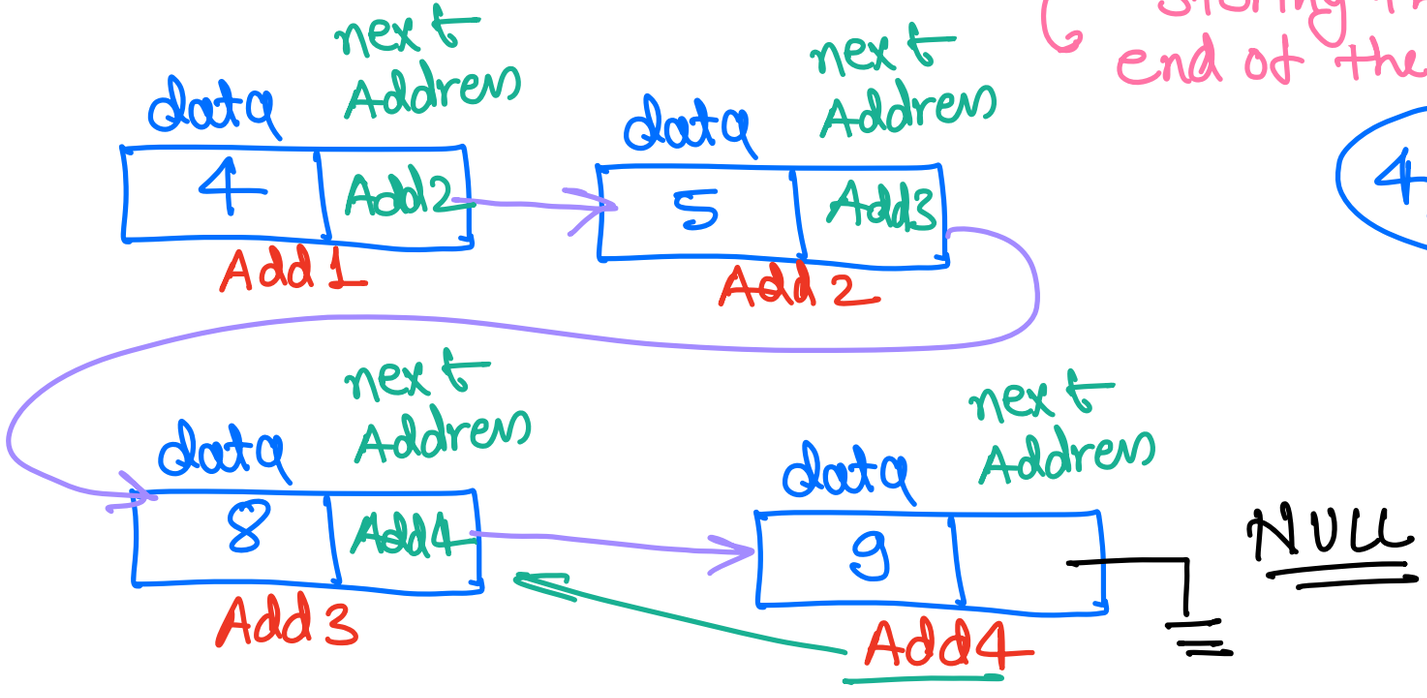
How much space we need to store the addres in this example. 1char, 1num

1Byte, 4Byte



denotes the end of storing the elements end of the list

Objn:



4, 5, 8, 9

§. storing int as array

```
int[] array;          // to integers.
```

```
class Node {
    int data;
    // address of next node;
    Node next;        // will store the address of the datatype "Node"
}
```

10:49

§. Pros & cons of arrays & linked list

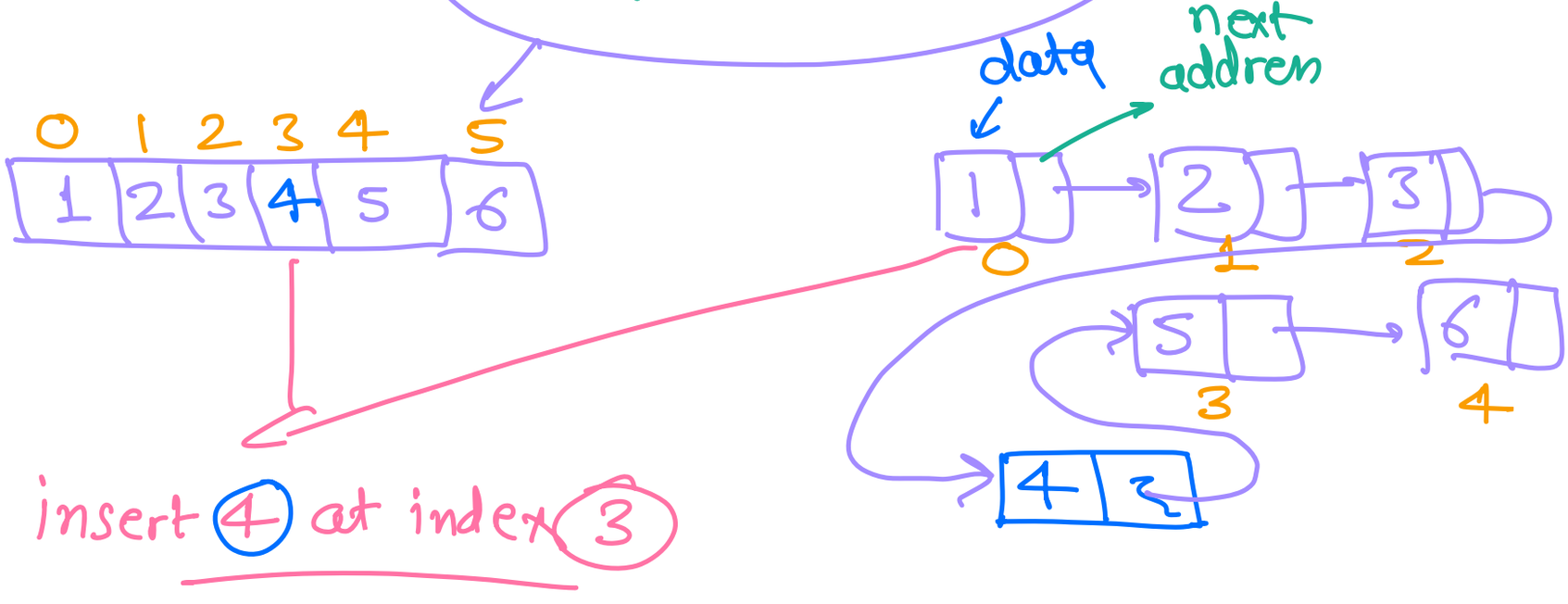
Array

+ Access any element in O(1) time
+ easy to implement
+ less memory than LL

— fixed length
— insertion & deletion are expensive

+ iterating is easy

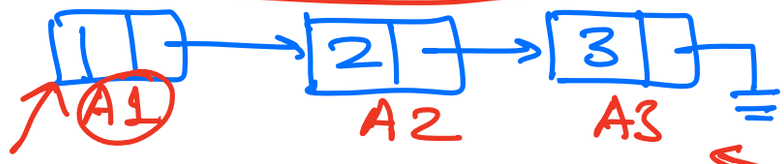
Linked list + can grow indefinitely − extra space req^d for addren
+ use fragmented memory well
+ insertion & deletion are easy compared to arrays



Q. Given a number N, create a linked list with $N \geq 1$ N nodes having data 1, 2, 3 ... N.
Return the head of the linked list

Address of the first node

ex: N = 3,



addresses

Obsn: always dry run your code for linked list problems.

idea: ① we need to create custom data type which can store the data (int) and the addren for next node. 11:05

② How to initialize the objects

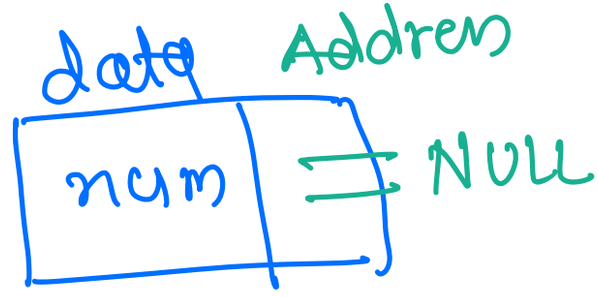
③ How to connect them.

④ return the starting node

if N == 0 ⇒ no link list ⇒ null

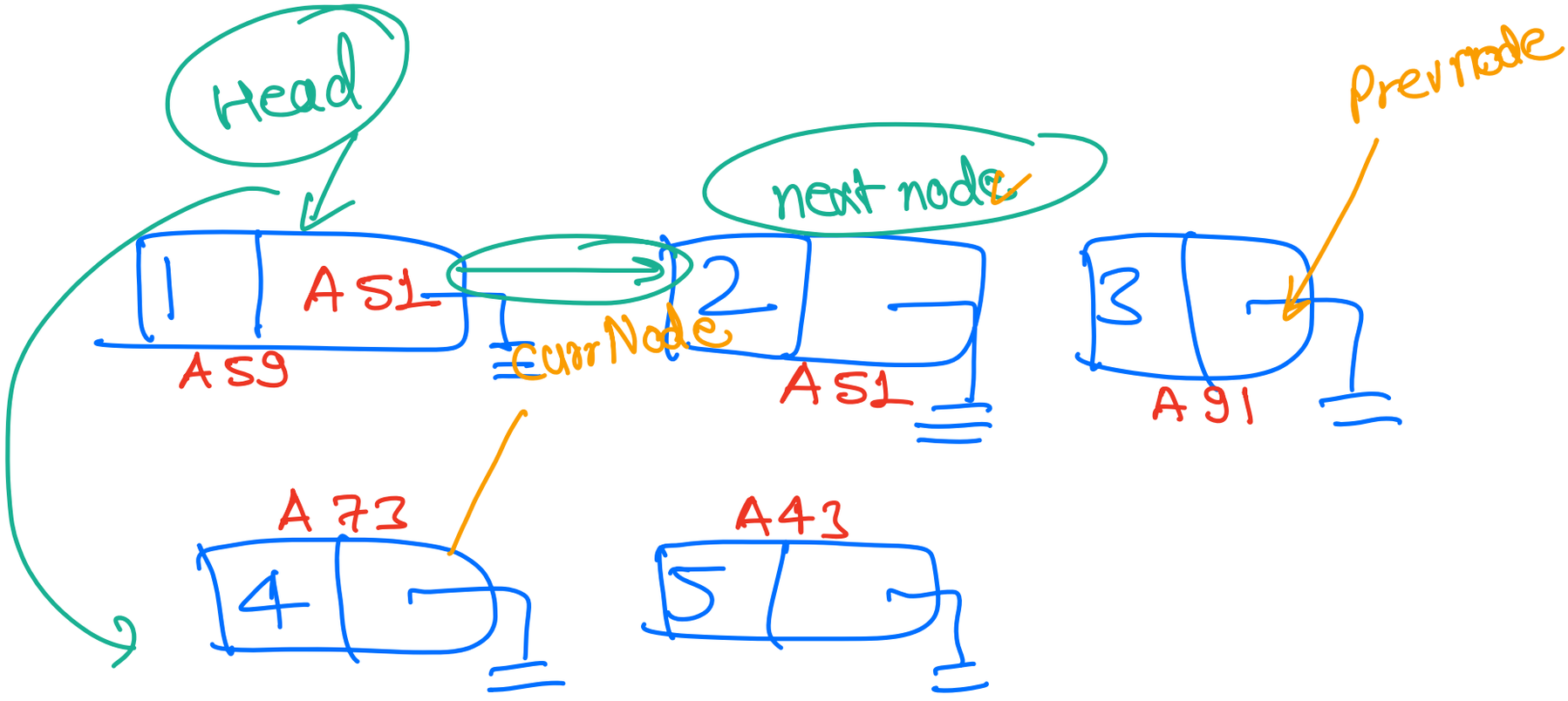
1

```
class Node {
    int data;
    // address of next node;
    Node next;
    Node (int num){
      this. data = num;
      this. next = null;
    }
}
```



// in Ctf⊙X


```
Node mynode
  = new Node(11);
```



currentNode.next = nextNode

```
class Node {
    int data;
```

```
    // address of next node;
        Node next;

        Node (int num){
          this. data = num:
          this. next = null;
        }
}

Node createLinkedList (int N){
      // we want to create the first node outside
         for loop

        Node head = new Node (1);
        Node prevnode = head:

        for (i=2; i<=N; i++)      {
          |   Node currentNode. = new Node(i);
```

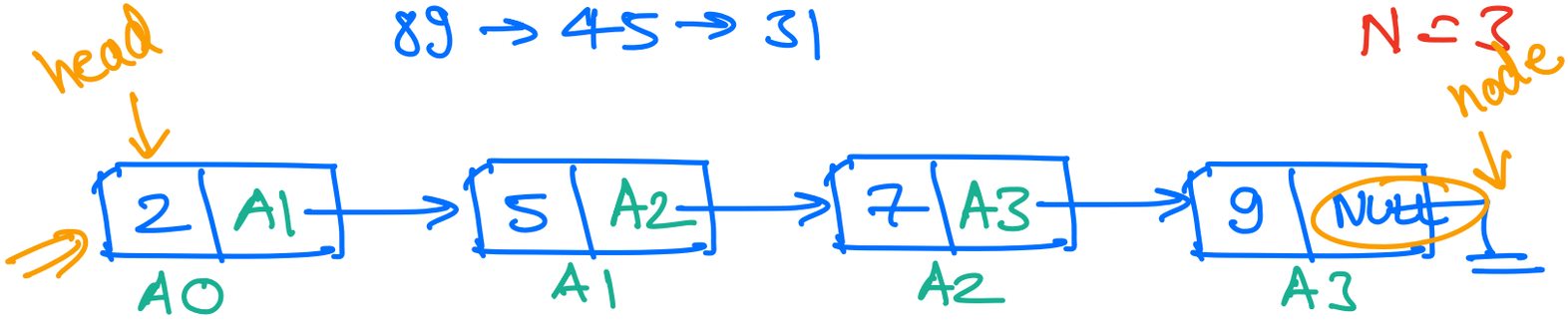
```
        prevNode.next = currentNode;
        prevNode = currentNode; //for next iterati
    }
    return head;
}
```

T.C. = O(N)
S.C. = O(N)

Q. print the size of the link list, given head Node.

ex. 1 → 2 → 3 → 11 → 12 → 15 N = 6

89 → 45 → 31 N = 3




```
int lengthofList(Node head) {
```

int i

```
Node node = head
int count = 0;

while (node != null) {
    count++;
    node = node.next;
}

return count;
}
//
```

count 0 1 2 3 4

```
while (node.next != null) {
```

H.W.

Q: print the linked list, given the headnode.

```
void printList (Node head) {

    Node node = head

    while (node != null) {
```

```
        print (node.data)
        node = node.next;
    }
    return;
}
```

Quiz 1: to travel the complete list we need first node

Doubt — playlist — linked list

→ Unlimited office hours for life

→ Thursday 4pm - 5pm

Tree Basics