CS & IT ENGINEERING

Data structure and Programming

Introduction to Data Structure

(In One Shot)

By- Pankaj Sharma sir

TOPICS TO BE COVERED

Introduction to Data Structure-1

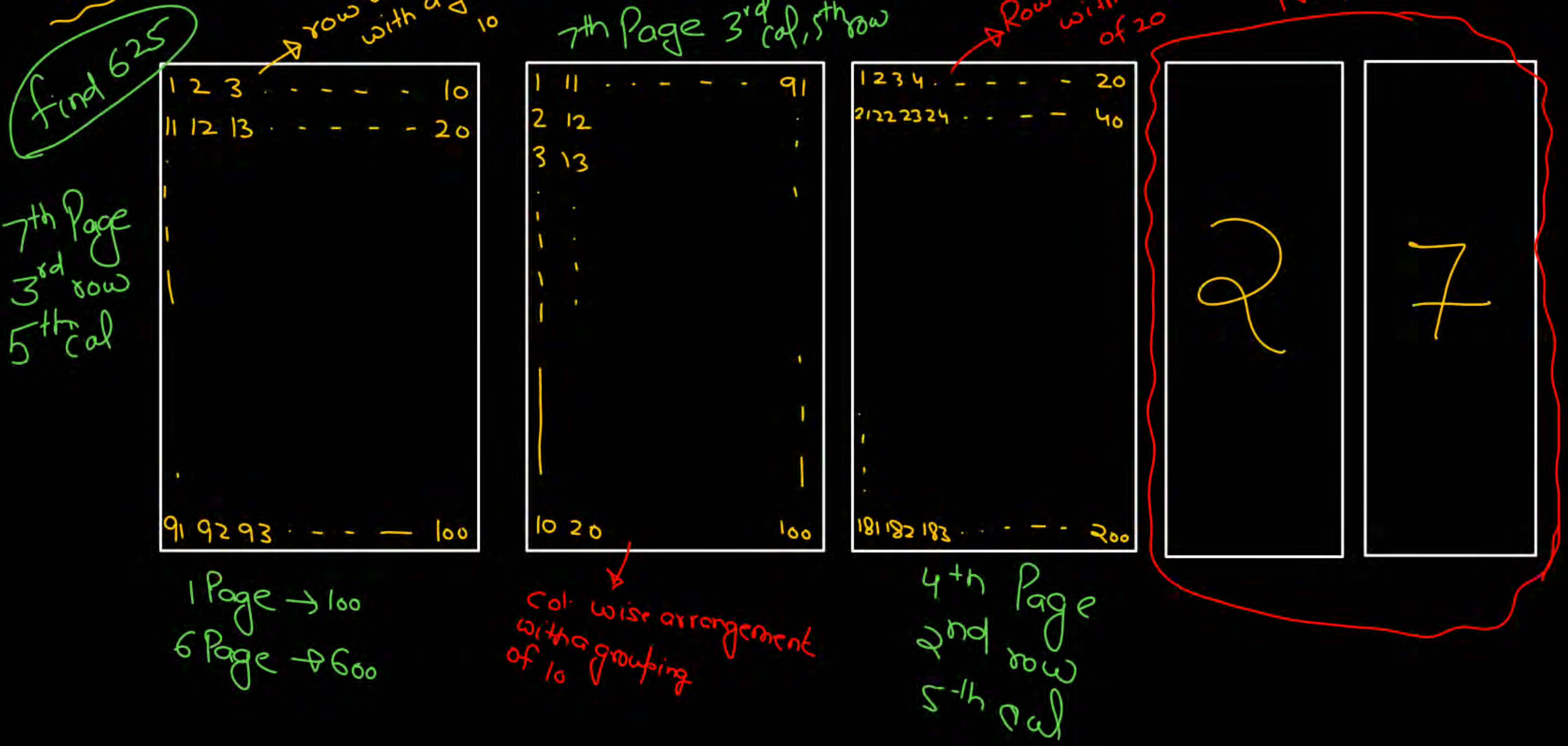
C Programming

- → Arrays ✓
- → Pointer ✓
- → structure ✓

DSA

- → placement
- → Gate

Data structure



1 to 1000

find 625

→ row wise arrangement with a grouping of 10

7th Page
3rd row
5th cal

1 Page → 100
6 Page → 600

7th Page 3rd col, 5th row

Col wise arrangement with a grouping of 10

→ Row wise arrangement with a grouping of 20

4th Page
2nd row
5th col

No pattern

Ex2

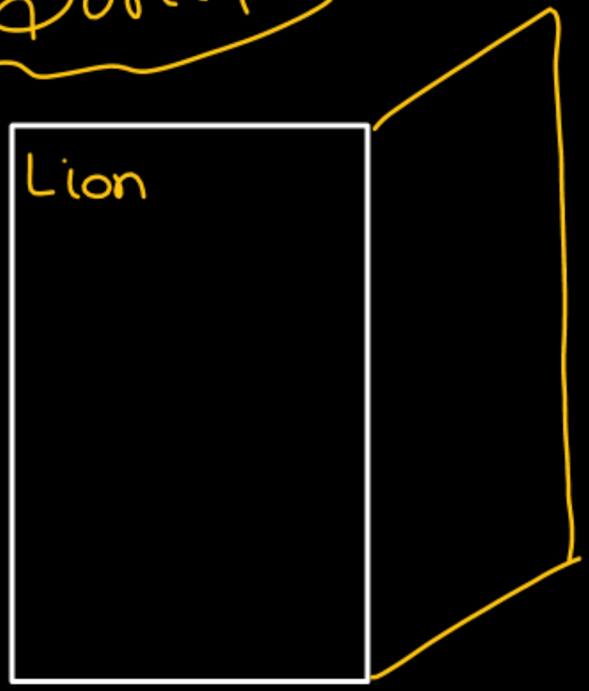
contact list ✓ ~~sorted~~ unsorted (Randomly)

~~search~~ 🔍

Ex3

Dictionary : Sorted

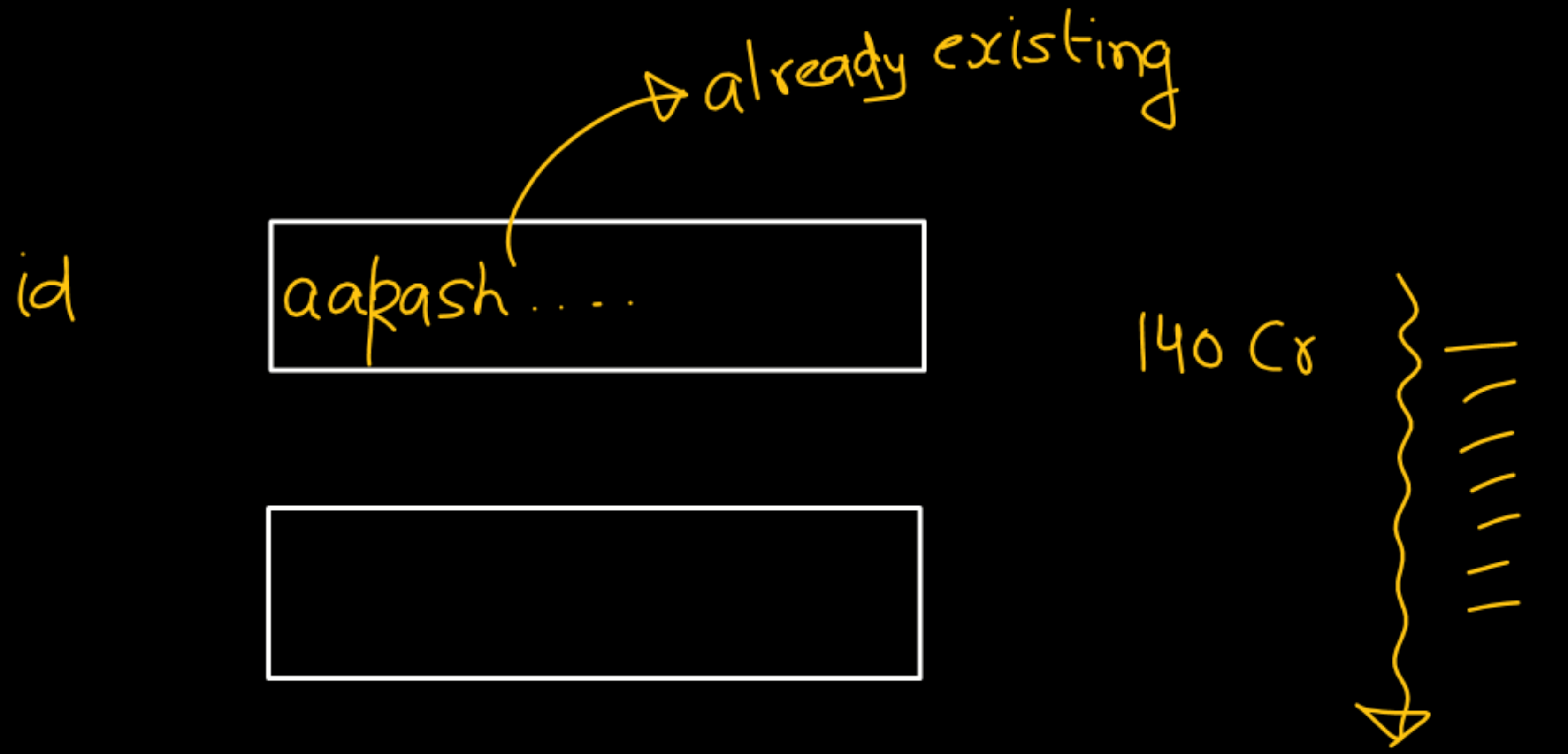
Parrot



Dictionary : Unsorted

Ex4

gmail ⇒ id



1. Contact list ⇒ Not Search

Does it matters ? → sorted
→ unsorted

2. Dictionary : ① Adding new words

and no one search for a word in dictionary.

Data Structure

1.) Linear data structure

An element can have atmost 2 neighbours.

2.) Non-linear data structure

It is possible that an element can have more than 2 neighbours.

Data structure

Linear data structure

(i) Arrays → address related Question

(ii) Linked list → code

(iii) stack

(iv) Queues

Non-Linear data structure

(i) Tree → trees
binary tree
binary Search tree
Heap
AVL - tree

(ii) Graph → * Algo

Hashing ✓ → almost Every year

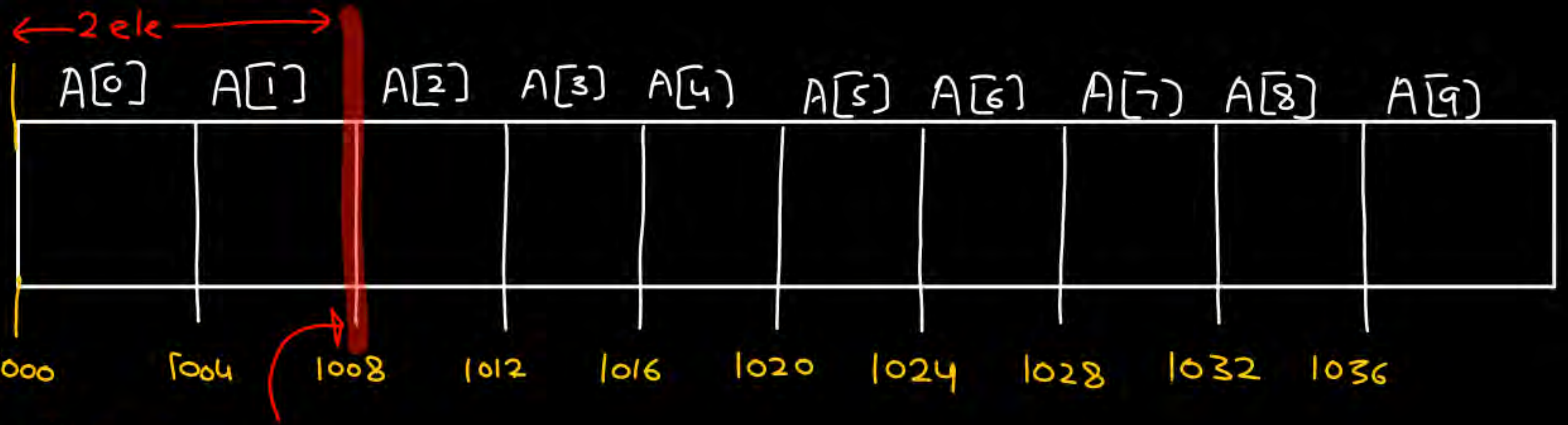
Array

- Collection of similar type of elements.
- Index starts from 0 (practically)
- Elements are stored one after another.
- Relative addressing
- Random access → access the element in constant time.
- Cache Friendlyness → i will cover today/ 1st chapter

int → 4 byte

b.A ⇒ 1000

int A[10]



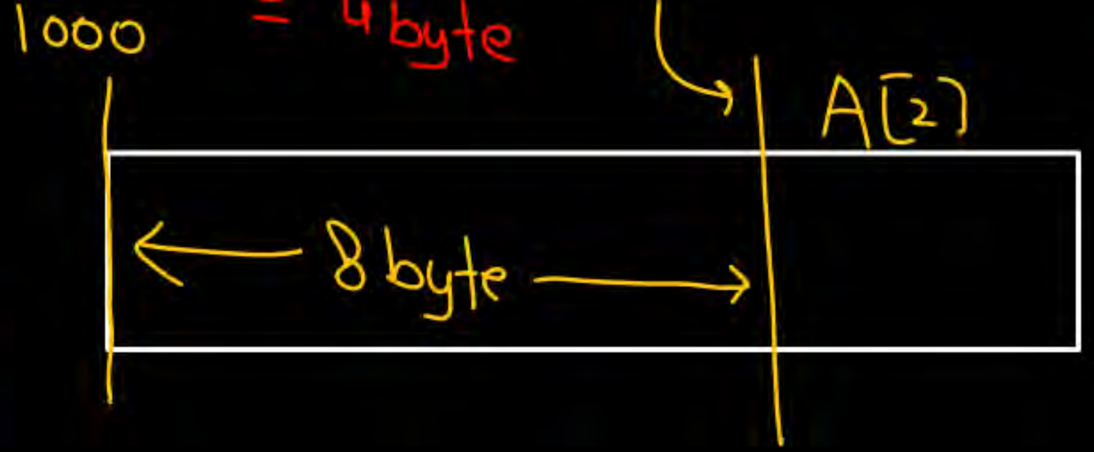
Address of A[0] = 1000

Address of A[2] = ?

① How many elements are already filled before A[2] = 2

② size of each ele = 4 byte

③ Memory already filled before A[2] = 2x4 = 8 bytes

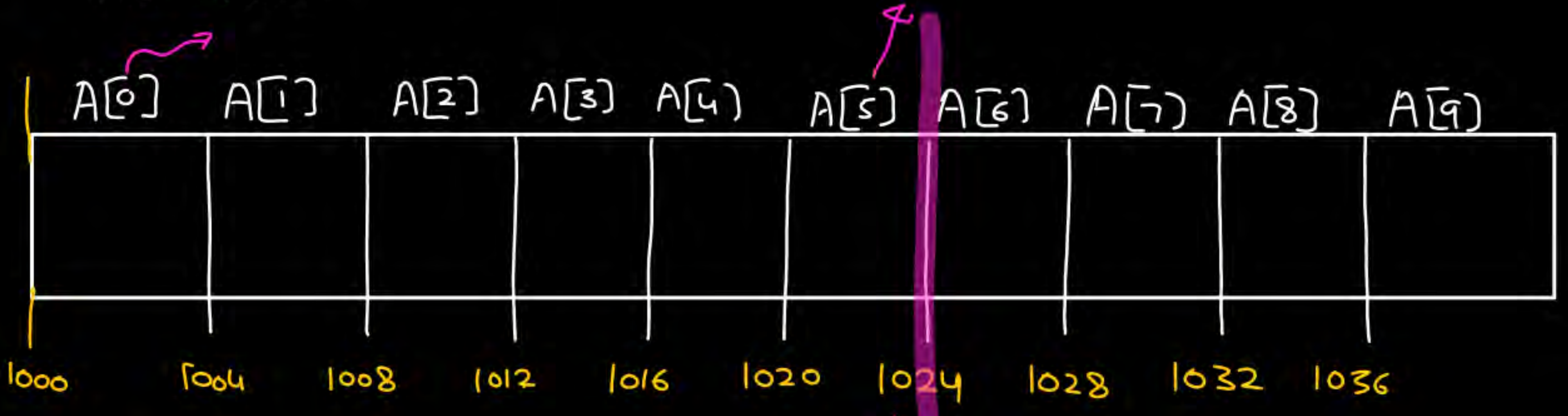


add(A[2]) = 1000 + 8 = 1008

int → 4 byte

b.A ⇒ 1000

int A[10]



Address of A[0] = 1000

Address of A[6] = ?

① How many elements are already filled before A[6]

= index 0 to 5

= 5-0+1 = 6

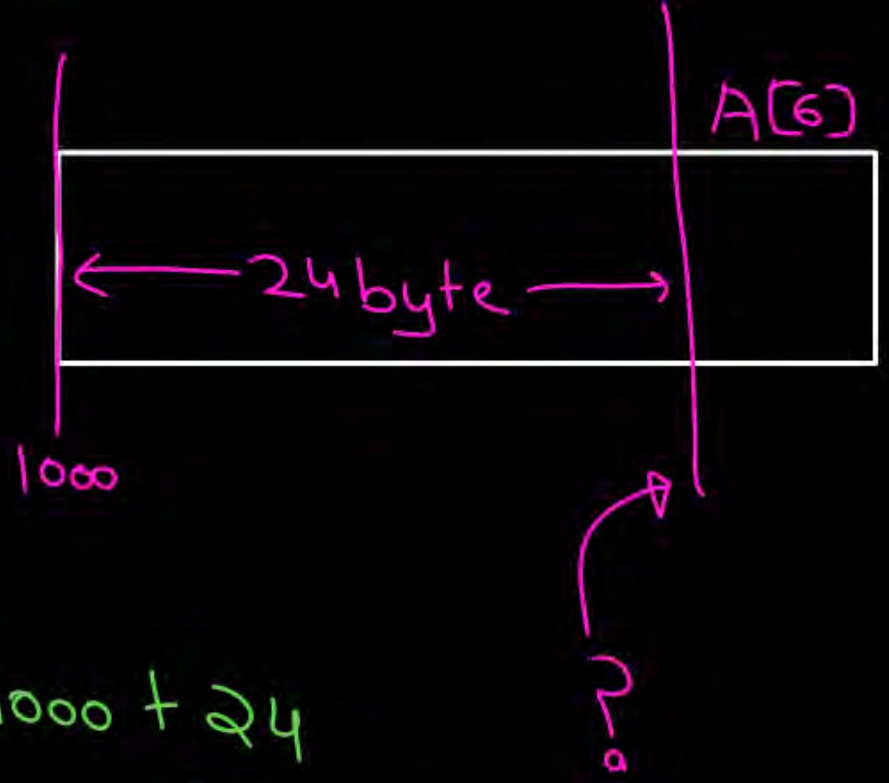
② Memory already filled before A[6]

= 6×4

= 24 byte

⇒ add (A[6]) = 1000 + 24

= 1024



In theory ⟹ index can start from any no.

$A[-5..5]$

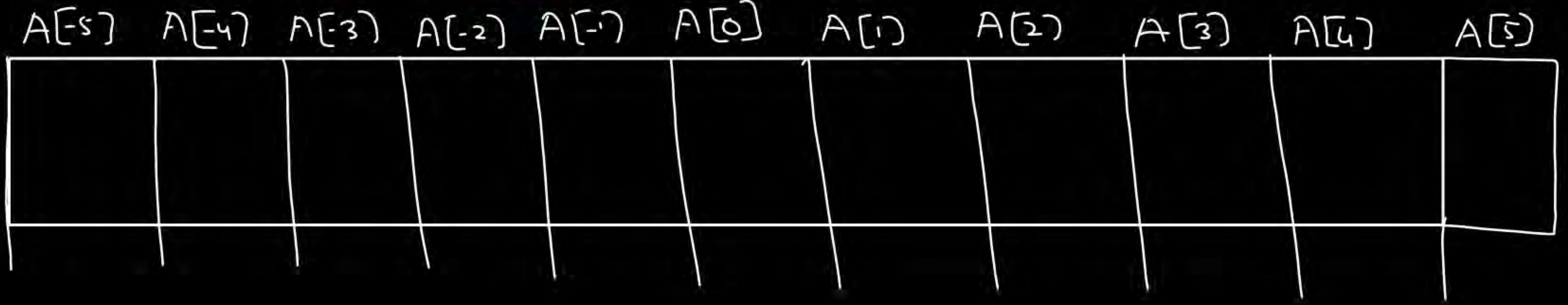
lowest index

Largest index

# of element = last - fir + 1

= $5-(-5)+1$

= 11 elements



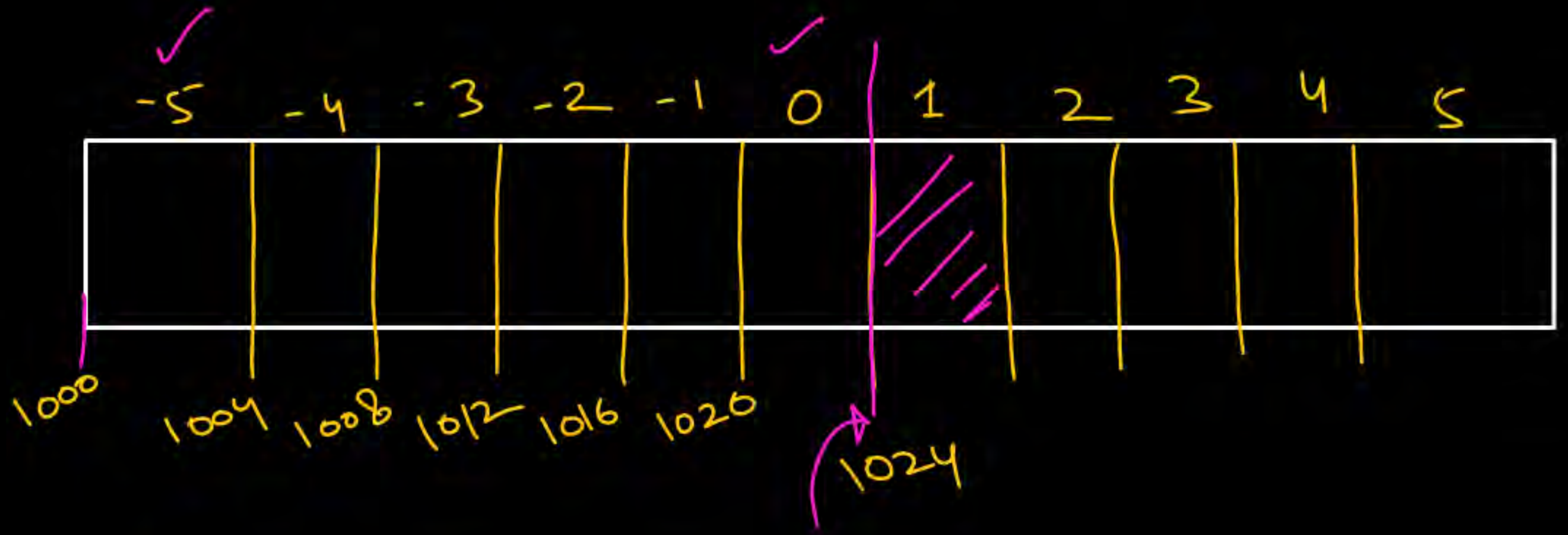
Q

$A[-5..5]$

$w = 4$ byte

Base address = 1000

$add(A[1]) = ?$



① How many elements already filled before $A[1]$ = index -5 to 0

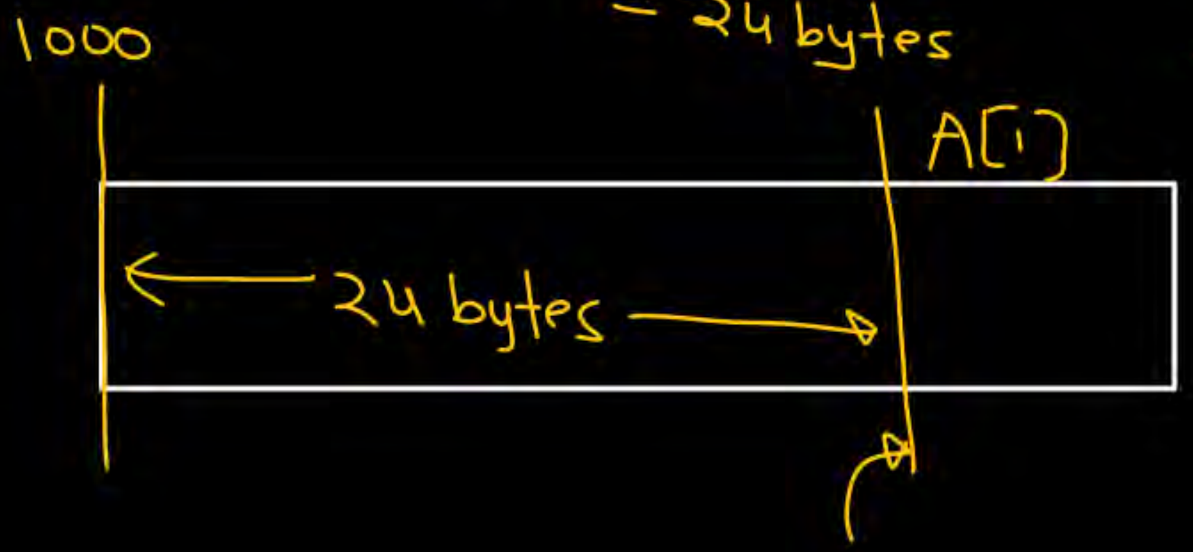
$= 0-(-5)+1$

$= 6$ elements

② Memory already filled before $A[1] = 6 \times 4$

$= 24$ bytes

$\Rightarrow 1024$



Q

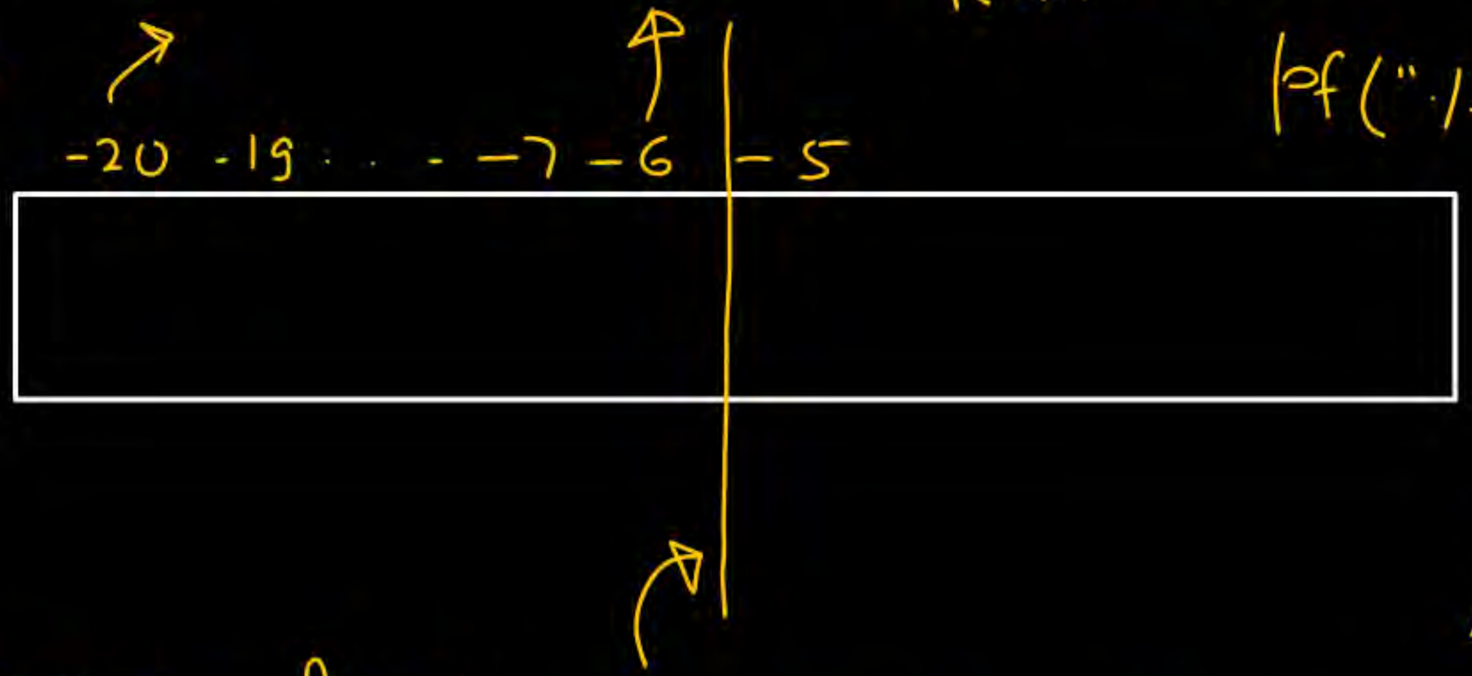
$A[-20..10]$

$w = 2$ byte

$BA = 1000$

$add(A[-5]) = ?$

$\boxed{1030}$



Random access

pf("%d", A[6]);

↓

Address

↓

constant time

① How many elements are already filled before A[-5]

$= -20$ to -6

$= -6-(-20)+1$

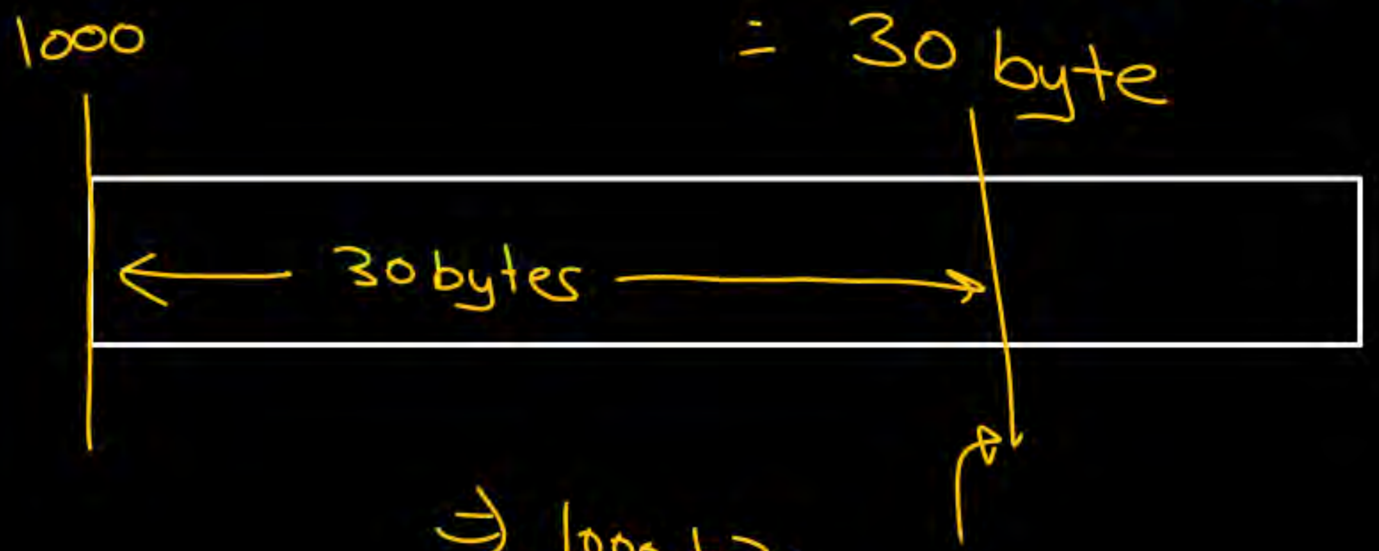
$= -6+20+1$

$= 15$ elements

② Memory already filled before A[-5]

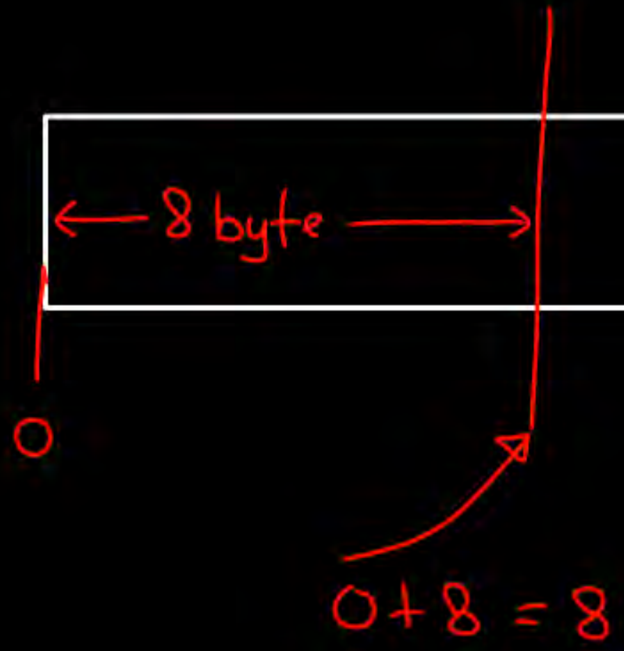
$= 15 \times 2$ byte

$= 30$ byte



$\Rightarrow 1000+30$

$= 1030$

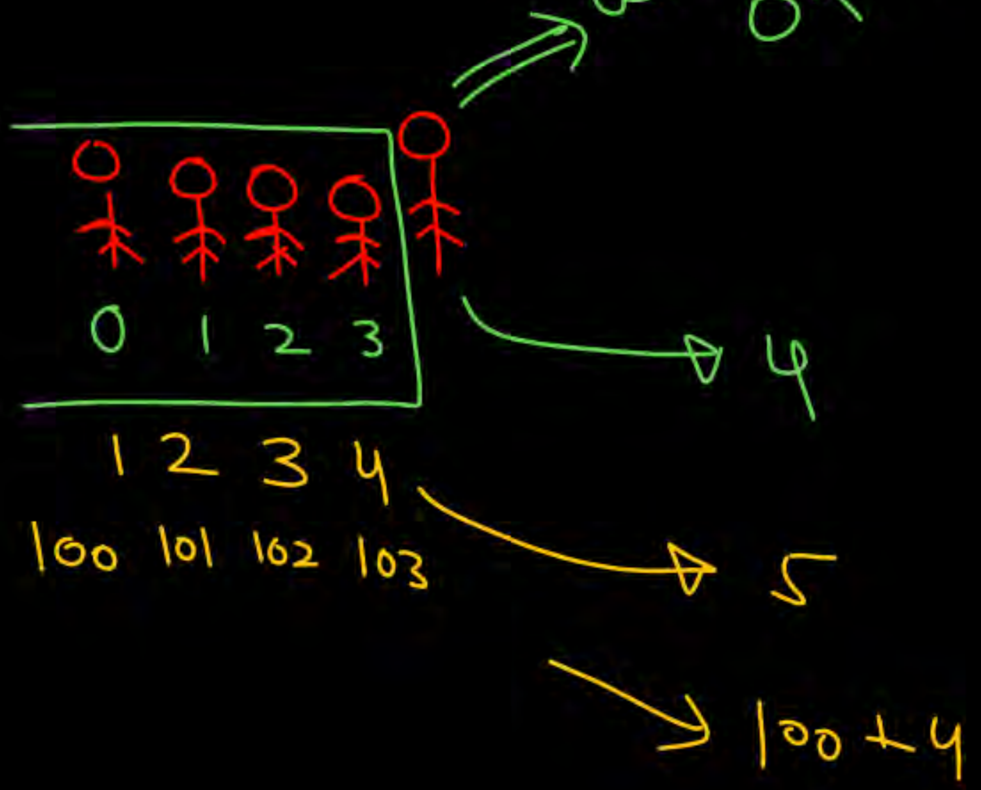


100 to 200 (including both)

⇒ 200 − 100 + 1 = $\boxed{101}$

last − first + 1

what is the no. of this student



2-D arrays

- * Revise - 15 min
- * 1-D array
- ✓ -ve index ✓

Theory

08 : 30 PM

P → is a pointer

$P-1$

$*(P-1)$

$P[-1]$

THANK YOU GW SOLDIERS !