Arrays....

PAGE No. 1
DATE / /

* Why Array?

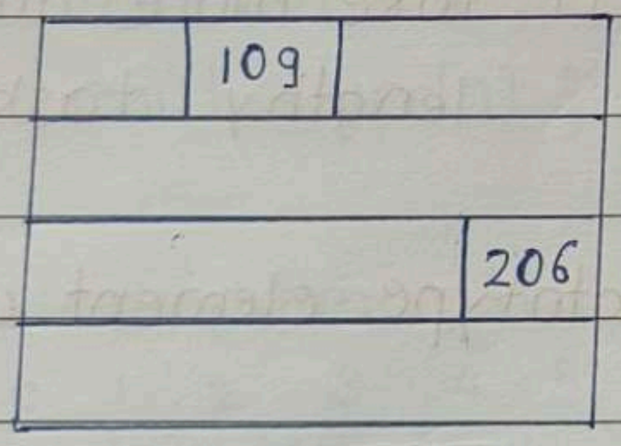
- A person want to store Apple, he rent a storeroom in nearby building. Ex: ROOM No: 109.

- The person have coming more Apple, he rent one more storeroom is same building.
  Ex: Room No: 206

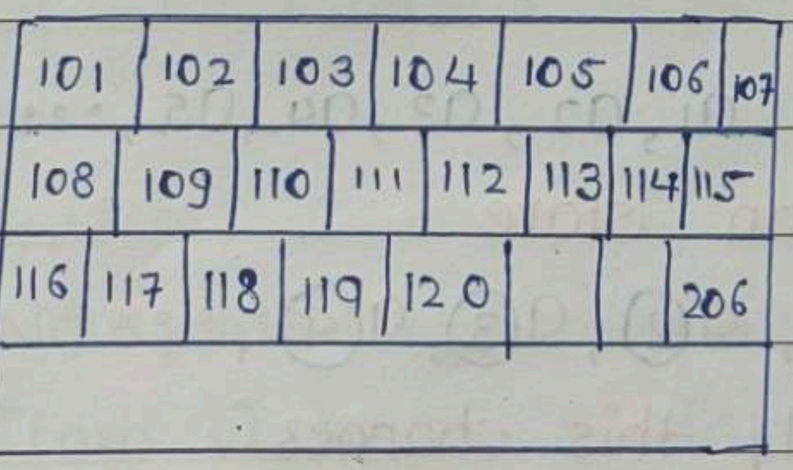
- The sale of apple is increased rapidly. He then rent 20 room in same building. But he want the room which are all continous.

- The owner has alloted room no. 101 to 120 to the person.

Previously -



In this, person has to remember the room number.



In this, person has to remember the room no. of 1st room only and rest of the room are continous.

Array : Collection of same datatype

Why we need ?

Example : • Sum of 3 no.

int a = 3
int b = 10
int c = 13

Easily we do

• Sum of 10 number

int a, b, c, d, e, f ... Easily we do

• Sum of 100 number

Make 100 variable takes some time and
then sum little bit lengthy

• Sum of 1000 number

Make 1000 variable It take more time and very
lengthy task

So for storing the same datatype element we are
array.

For storing 1000 no. : int a1, a2, a3, a4, a5, ...

By this way we can store

By seeing deeply only : a①, a②, a③, a④, ...

↳ only this changes

So, we can done this in another way by array

a[i] ⟶ a[0], a[1], a[2], ...

Array is a collection of same datatype.
It is stored in contigous location

- Type of element is same in array

arr[] = type → int / → float / → char

Declaration : int arr[7]; → It makes 7 space in memory to store integer

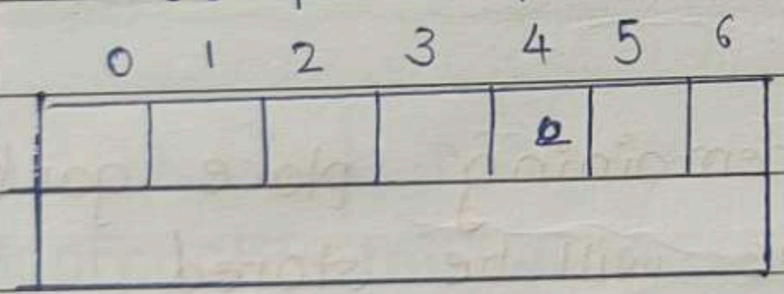
(int: datatype, arr: name, 7: size)

0	1	2	3	4	5	6
9	2	6	0	-3	18	26

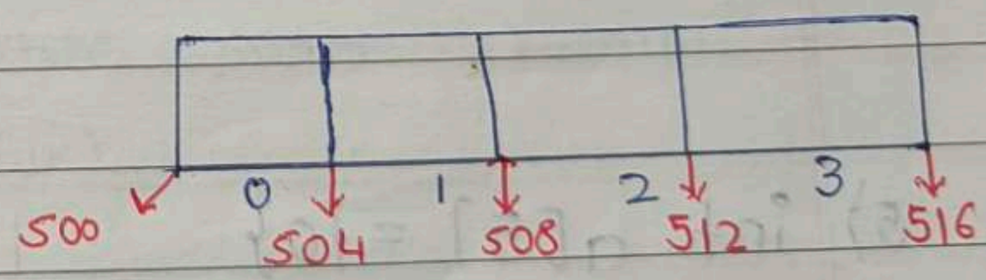
* Why indexing is start from 0?

```
arr[0] = 9;
cout<<arr[0]; //prints 9
```

In computer, it uses byte addressable



⟹ Integer - type arrays

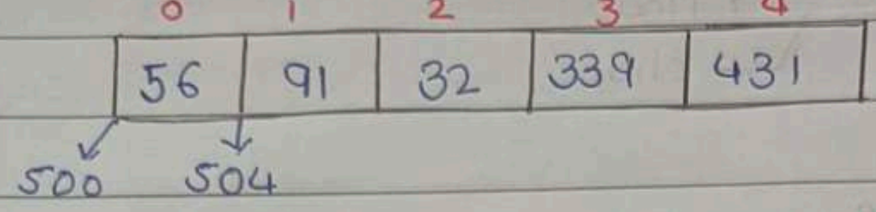


Now, We use indexing from 0 because for accessing the address in 1-based indexing can take more time.

In 1-based indexing :
Initial address + (index - 1) * size , → more operation

In 0-based indexing :
Initial address + index * size , → less operation

int arr[5];



Integer ~~can~~ takes 4 byte to store an integer

Different types of initialization in array :

1) int a[5] = {6, 8, 5, 1, 9}

2) int name[] = {3, 8, 2, 9}

3) int arr[10];
for (int i = 0 ; i<10 ; i++)
cin >> arr[i]; // values in arr is taken by user

4) int a[5] = {8, 4} // at remaining place garbage value will be stored

5) int a[5] = {0} // all elements will be stored as zero ; no other value will be stored.

Printing of array -

```
int arr[5] = {0,10,7,11,8}
    for (int i =0 ; i<5 ; i++)
        cout << arr[i];
```

Que-1) Find Minimum element :

Ex: arr[5] = {4,6,11,2,8}

Minimum element : 2

0	1	2	3	4
4	6	11	2	8

Approach

- Initialize a variable with INT-MAX (because we have to find smallest value)
- Traverse the array and update the var if small no. exists after comparing.

Psuedocode :

```
//MIN value
for (int i=0 ; i<5 ; i++) {
    if (arr[i] < ans)
        ans = arr[i];
}
cout << ans << endl;
```

Problem - 2) Find Maximum Element :

Approach - Initialize a variable with INT-MIN (becoz we have to find largest value)

Psuedocode :

```
//MAX VALUE
int ans = INT-MIN;
for (int i= 0 ; i<5 ; i++) {
    if (arr[i] > ans) {
        ans = arr[i];
}
cout << ans << endl;
```

Problem-3 Find Second Smallest Element

Approach -
- Initialize two var : smallest, secsmallest with INT-MAX.
- Traverse the array and first find smallest
- Now traverse again and check whether current element is ~~greater than~~ equal to smallest and less than secsmallest.

Psuedocode :

```
int print2smallest (int arr[], int n) {
    int ans = ~~INT-MIN~~; INT-MAX;  smallest
    for (int i=0 ; i<n ; i++) {   // ~~largest~~ element
        if arr[i] <ans)
            ans = arr[i];
    }
    // Second smallest element
    int secsmallest = INT-MAX;
    for (int i=0 ; i<n ; i++) {
        if (arr[i] != ans) && arr[i] <secsmallest){
        secsmallest = arr[i];
    } }
    ~~return~~ return secsmallest;
```

We can use min function instead of arr[i] < secsm...

× ~~secsmallest~~ secsmallest = min(secsmallest, arr[i]

PAGE No. 4
DATE / /

Problem - 4 Find Second Largest Element

Approach -
- Initialize two var : largest , slargest with INT-MIN
- Traverse the array and first find s largest
- Now traverse again and check whether currrent element is not equal to largest and slargest compare slargest and current & store max in slargest

Psuedocode :

```
int print2largest (int arr[], int n) {
    int largest = INT-MIN;
      // largest element
    for (int i=0 ; i<n; i++){
        if (arr[i] > largest)
        largest = largest [i];
}
    // second largest element
    int slargest = INT-MIN;
    for (int i=0 ; i<n ; i++) {
        if (arr[i] != slargest)
        slargest = max(slargest, arr[i];
}

    return slargest;
}
```

Problem - 5 Search Element in Array (Linear search)

Ex : arr[] = {1,2,3,4,5 key = 3

Output = 2

Approach -

- We will traverse the whole array and see if the element is present in the array or not.
- If found we will print the index of element otherwise we will print -1.

Psuedocode -

```
int search (int arr[], int n, int key) {
    for (int i = 0 ; i < n ; i++) {
        if (arr[i] == num)
            return i;
    }
return -1;
}
```

Problem-6 Reverse of an Array

Ex : arr[] = {1,2,3,4,5,6} ⇒ Output : 6,5,4,3,2,1

Approach 1 - Traverse the array and print array from the last

Psuedocode -

```
for (int i = arr.length -1 ; i >= 0 ; i--) {
        cout << arr[i] << " ";
    }
```

Approach 2 - We can create another array of same length and then store elements from last to first and ther restore it in first but space will be

consumed more so we will not use it.

~~Approach 3~~ - Psuedocode - (Approach-3)

```
int reverse (int arr[], int n) {
    int start = 0 , end = n-1 ;
    while (start < end) {
        swap(arr[start], arr[end]);
        start ++ ;
        end -- ;
    }
    for(int i = 0 ; i < n ; i++)
        cout << arr[i] << " " ;
}
```

Approach 3 : We have to swap elements from index a[n-1] to a[0] and so on... and the terminating condition will be if start index is greater than end index then the loop will terminate.

Problem-7 Find Fibonacci Series using Array

Approach : We know that fibonacci series has first two term predefined : a[0] = 0 , a[1] and the next element is sum of previous two elements.

0, 1, 1, 2, 3, 5, 8, 13, 21, ...

Psuedocode :

```
int n ;
cin >> n ;
int arr[1000] ;
arr[0] = 0 ;
```

```
    arr[1] = 1;

    for (int i = 2 ; i<=n-1 ; i++ ) {
        arr[i] = arr[i-1] + arr[i-2];
}   cout << arr[n-1];
}
```

Problem-8 : Missing Number

Ex : arr[] = {1, 3, 4, 5, 6} ; n=5 ; missing No : 2
as array should contain 1 to n

Approach -

- We will find sum of no. from 1 to using (n(n-1)) /2 .
- Then we will find sum of array and then we will subtract them and return the sub value.

Psuedocode -

```
int missingNumber (int n, vector <int> &arr) {
    int sum = 0;
    int ans = 0;
    sum = (n*(n+1)) /2 ;
    for (int i = 0 ; i<n-1 ; i++) {
        ans += arr[i] ;
    }
    return sum -ans;
};
```

Problem - 9 : Rotate 1 place of Array

Ex: arr = { 2, 3, 7, 11, 4 }

output = { 4, 2, 3, 7, 11 }

Process : 2 3 7 11 4 int num = 4

arr[i] = arr[i-1]

2 3 7 11 11

arr[i] = arr[i-1]

2 3 7 7 11

2 3 3 7 11

2 2 3 7 11

arr[0] = num

Ans : 4 2 3 7 11

Approach : Copy the last element in temporary variable, using for loop add the previous value in next index and at end replace a[0] with temp variable.

Psuedocode :

```
void rotate (int arr[], int n) {
    int last = arr[n-1];
    For (int i = n-2 ; i >= 0 ; i--) {
    } arr[i+1] = arr[i];
    arr[0] = last ;
}
```

PAGE No.
DATE / /

Problem -10 : Find Duplicates

Approach -

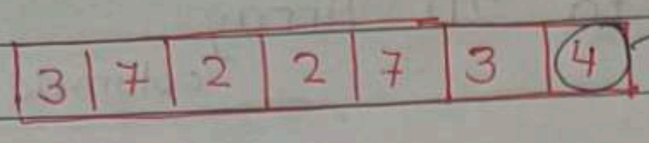
Psuedocode -

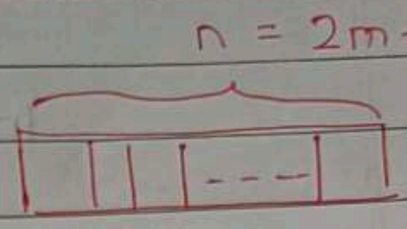
```
int findDuplicate (vector <int> & arr) {
    int ans = 0;
    for (int i = 0 ; i < arr.size() ; i++) {
        ans = ans ^ arr[i];
    }
    for (int i = 1 ; i < arr.size() ; i++) {
        ans = ans ^ i ;
    }
    return ans ;
}
```

and
from
2D
mod
LUV

PAGE No. 7
DATE / /

Problem - 11 : Find Unique Element





XOR : $a \wedge a = 0$

$0 \wedge a = a$

1 - no. appear once

Psuedocode :

```
int FindUnique (int arr[], int size) {
    int ans = 0;
    for (int i=0 ; i<size; i++) {
        ans = ans ^ arr[i];
    }
    return ans;
}
```

Problem - 12 : Swap Alternate

Example : i/p → arr[5] = {1, 2, 7, 8, 5}
o/p = {2, 1, 8, 7, 5}

i/p → {1, 2, 3, 4, 5, 6}
o/p = {2, 1, 4, 3, 6, 5}

Psuedocode :

```
void swapAlternate (int arr[], int size) {
    for (int i=0 ; i<size ; i=i+2) {
        if (i+1 < size) {
            swap (arr[i], arr[i+1]);
    } {
}
```

Modular
↳ All r

Prefix and
↳ fro
↳ 2D

STL m
↳ Lu