ARRAYS

Non Primitive datatypes

→ build using primitive data types

(1) ARRAYS

```
int m1 = 30;
int m2 = 40;
int m3 = 50;
```

marks

```
int marks[]; // Declaration
marks = new int[5]; // Initialisation
```

size of array // length

```
marks[0] = 30;   1st
marks[1] = 40;   2nd
marks[2] = 35;   3rd
marks[3] = 50    4th
marks[4] = 60    5th
```

→ marks of students

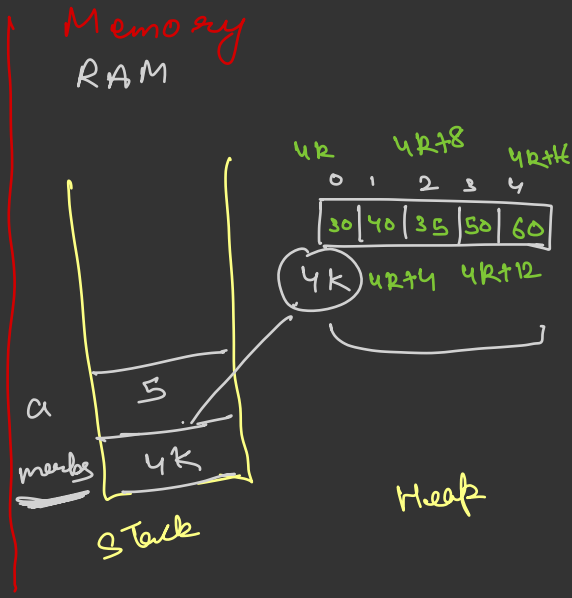
```
Print marks
for(int i=0; i < marks.length; i++){
    syso(marks[i]);
}
```

MEMORY MANAGEMENT IN ARRAY

```
int marks[]; // ①
marks = new int[5];
```

allocates the memory in heap

```
int a = 5;
```

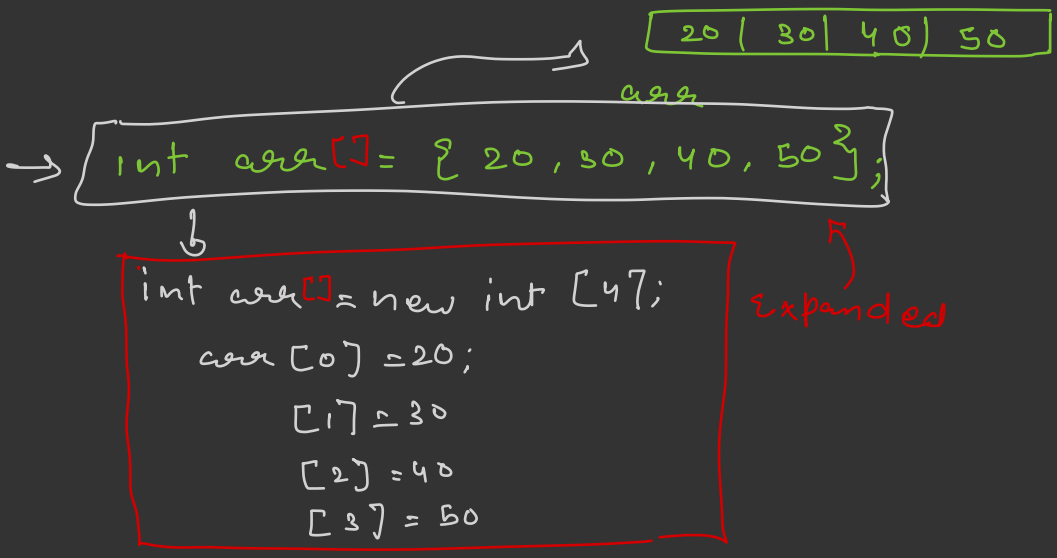



```
int arr[]; // int []arr;
```

Exactly Same

Boolean array

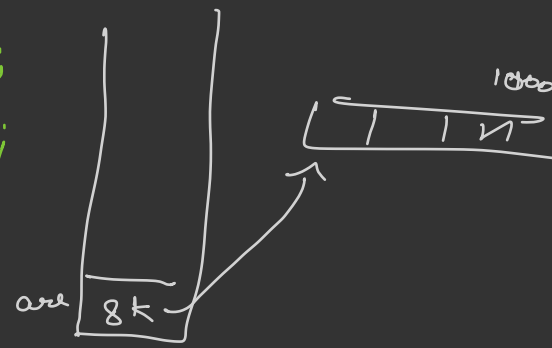
```
boolean arr[] = new boolean[10]; //
int arr[] = new int[5]; //
```



char arr[] = new char [2];

int arr = new int [1000];

Same time
- arr [500] = 20;
- arr [0] = 10;



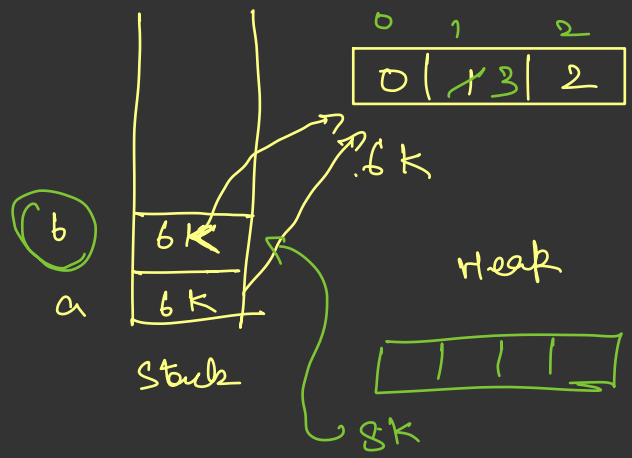
8K + 4 × 500

8K + 2K

= 10K

10000

```
int a[] = new int[3];
a[0] = 0;
a[1] = 1;
a[2] = 2;
int b[] = a;
b[1] = 3;
```



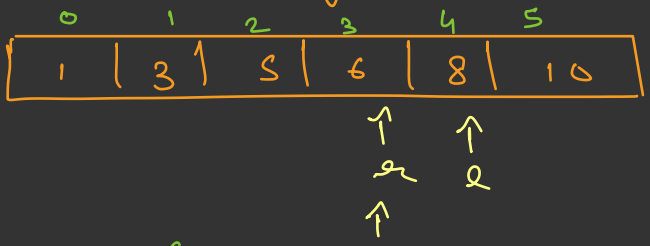
Print a] ⇒ 0 1 2

Print b] ⇒ 0 3 2]

⇒ b = new int[4];

Binary search

↳ Sorted array



val (7)

int m = (l + r) / 2

(0 + 5) / 2 = 5 / 2 = 2 // 3 + 5 / 2 = 4

mid = l + (r - l) / 2 →

```
if ( val == arr[mid] {
    syso mid;
}
else if ( val > arr[mid]){
    left = mid + 1;
else {
    right = mid - 1;
} }
```