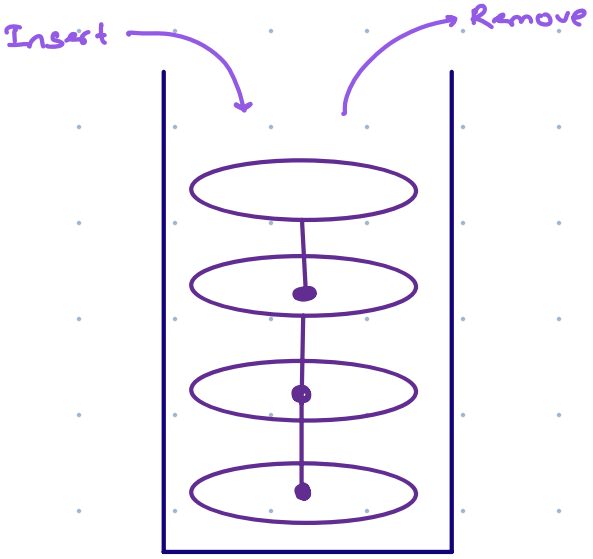
Memory Management

Today's content

01. Basics of Memory management
02. Function / call stack
03. Memory of basic data type
04. Memory of array & 2D array

* Quizzes

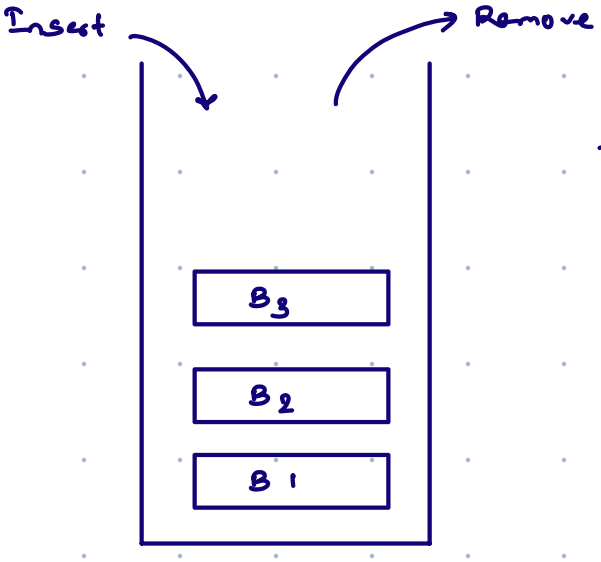
Idli cooker



→ Can remove & insert the plate from top

→ Access for my topmost plate.

Stack of Books



→ can remove & insert the books only from the top

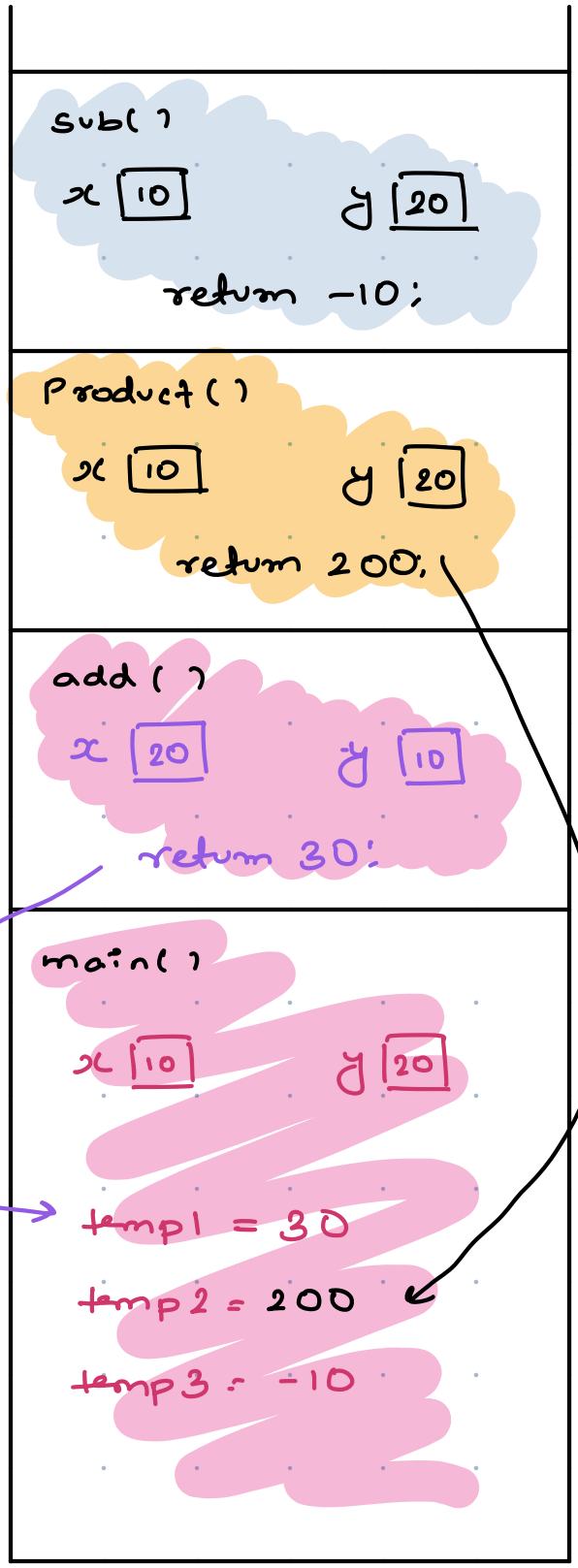
LIFO

Last In First Out

Call Stack

Function calls

```
static int add (int x, int y)
    return x+y
}

static int product (int x, int y)
    return x*y
}

static int sub (int x, int y)
    return x-y
}

public static void main ( ) {
    int x = 10;
    int y = 20;
    int temp1 = add (y, x);
    int temp2 = product (x,y)
    int temp3 = sub (x,y)

    SOP ( temp1 + temp2 + temp3) ;
}
```

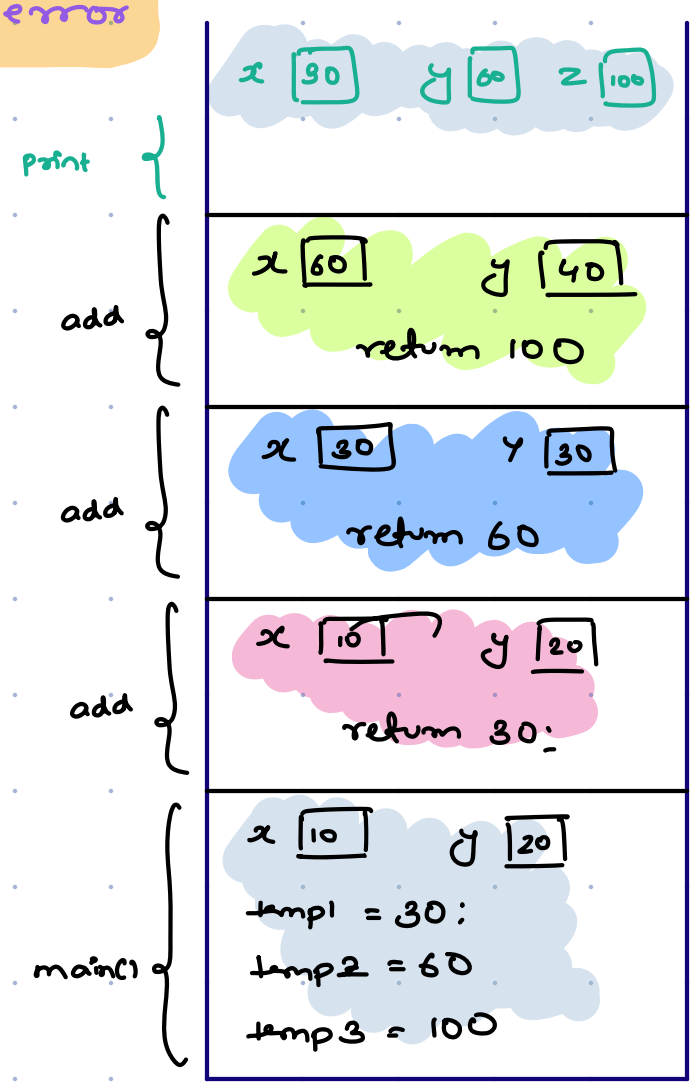


Output = 220

Note → We are only allowed with $\approx 10^5$ calls for our call stack. If your program gives more no. of calls, it will throw

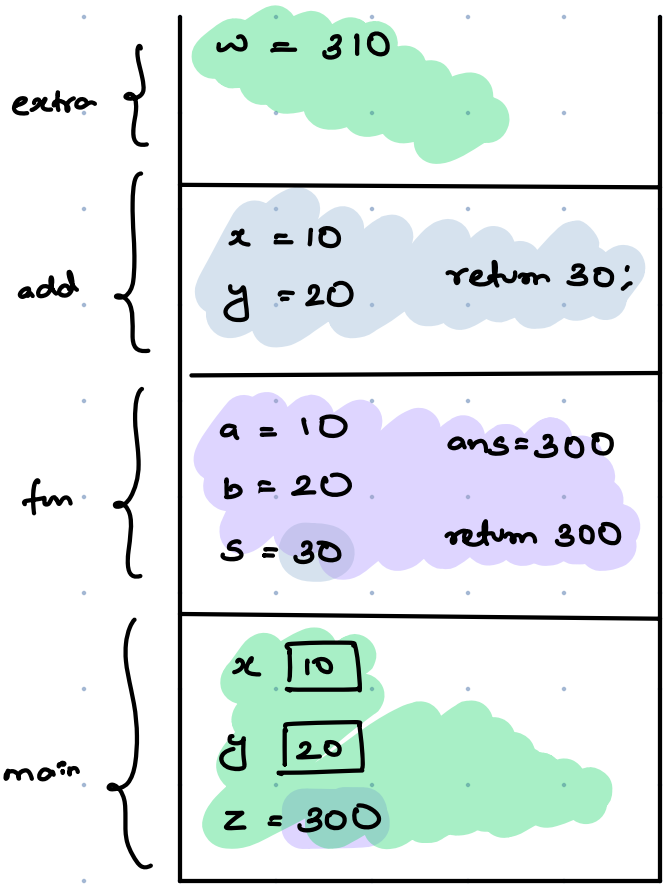
Stack overflow error

```
static int add (int x, int y)
{
    return x+y
}

public static void main ( ){
    int x = 10;
    int y = 20;
    int temp1 = add (x,y)
    int temp2 = add (temp1, 30);
    int temp3 = add (temp2 ,40);
    print (temp1, temp2, temp3)
}

static void print (int x, int y, int z){
    SOPln (x+y+z);
}
```



Output = 190

```
static int add (int x, int y) {
    return x+y;
}
static int fun (int a, int b) {
    int s = add(a,b);
    int ans = s*10;
    return ans;
}
static void extra (int w) {
    S.O.Pln ("Hello");
    S.O.Pln (w);
}
public static void main () {
    int x = 10;
    int y = 20;
    int z = fun (x,y);
    S.O.Pln (z);
    extra (z+10);
}
```



Output

300

Hello

310

* Types of Memory

01. Stack memory → Primitive datatypes variables (int, short, byte, long, char, boolean)

→ function calls

→ local variables

→ Object Reference / Object address

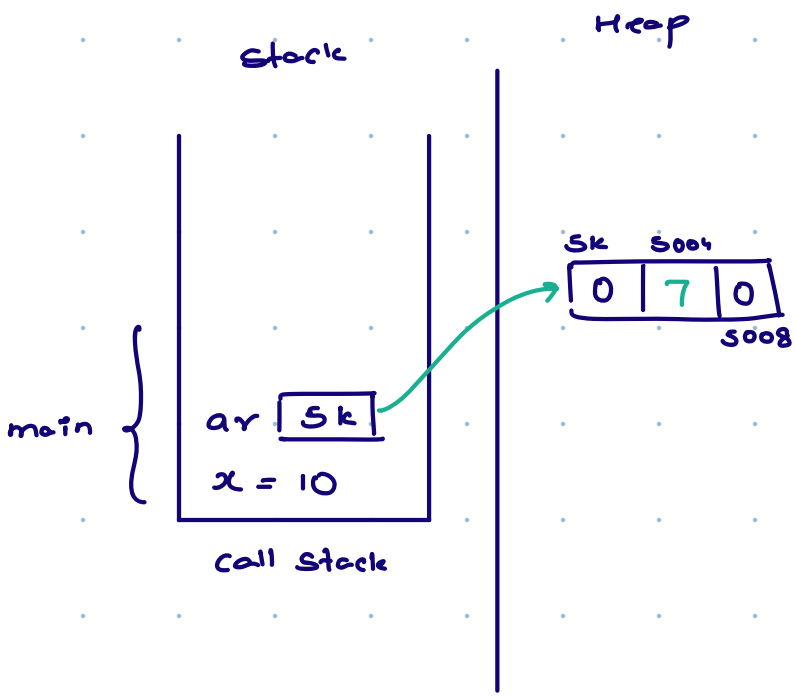
Or Heap memory → Container of the Object reference / real object / instance of class is created in heap memory.

Arrays. AL, String, Object

→ Static variables / Global variables

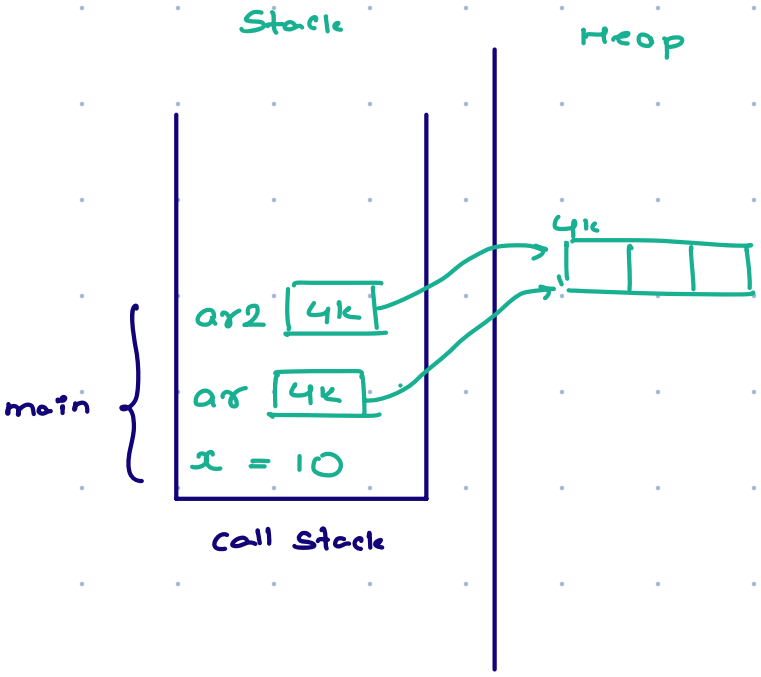
Ex 1

```
public static void main ( )
{
    int x = 10;
    int [ ] ar = new int [3];
    S.O.P (ar);  → 5k
    SOP ( ar[2] ) → 0
    ar[1] = 7;
}
```



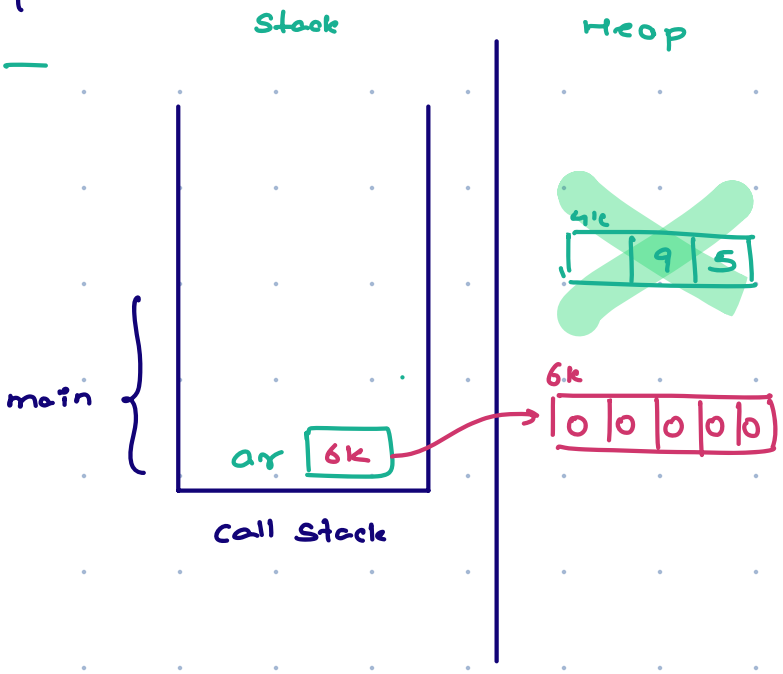
Ex 2

```
public static void main ( ){
    int x = 10;
    int [ ]ar = new int [3];
    int [ ]ar2 = ar ;
    S.O.P (ar); → 4K
    SOP ( ar2 ) → 4K
}
```



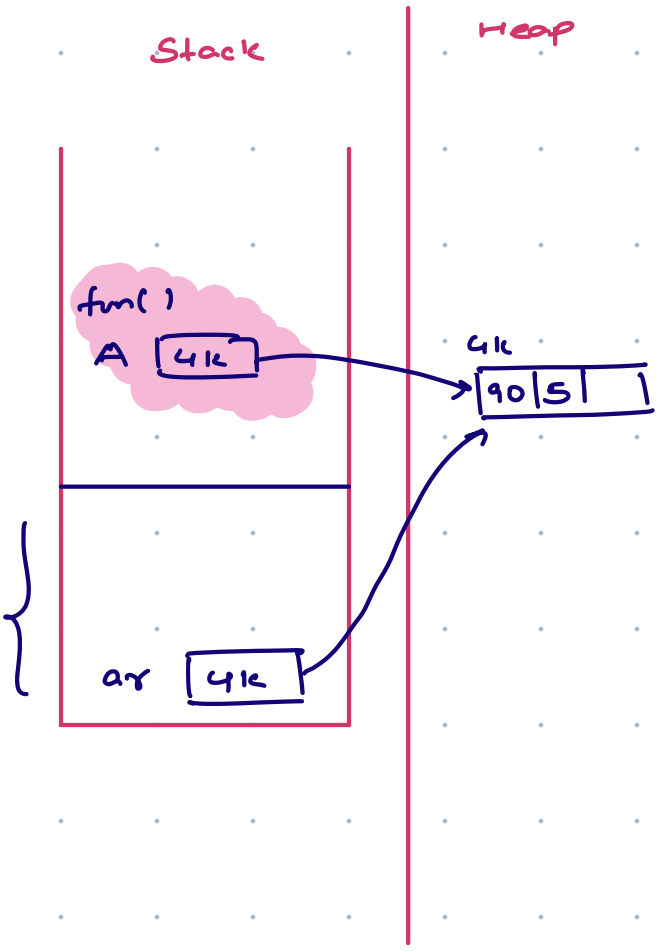
Ex 3

```
public static void main ( ){
    int [ ]ar = new int [3]; —
    S.O.P (ar); → 4K
    ar[1] = 9;
    ar[2] = 5;
    ar = new int [5];
    SOP ( ar[2] ) → 0
}
```



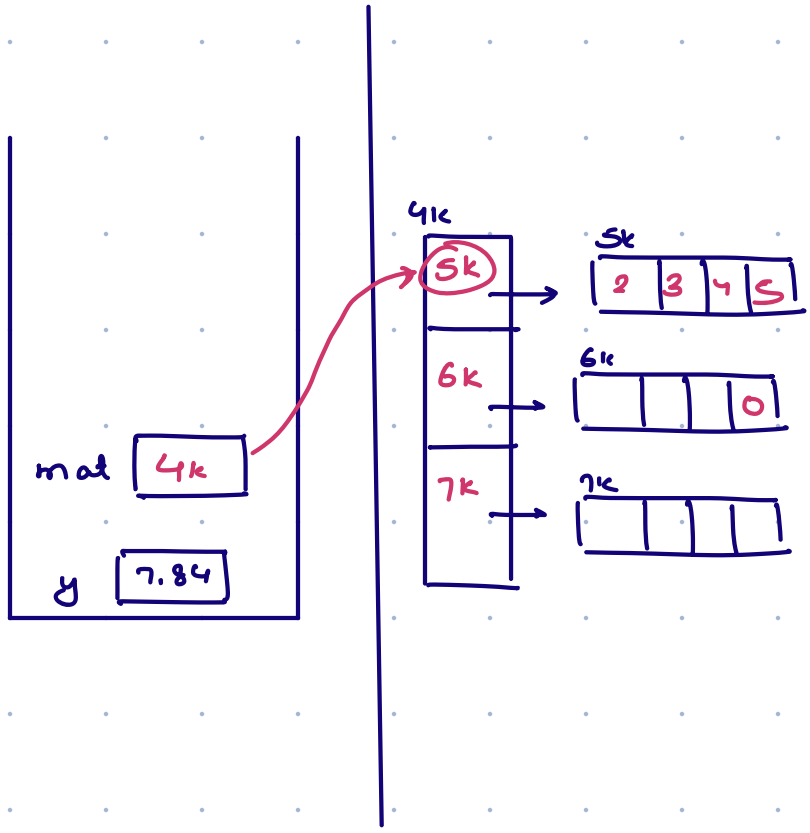
Ex 4

```
static void fun (int [] A)
{
    SOP (A) → 4k
    A[1] = 5;
}

Public static void main () {
    int [] ar = new int [3] ✓
    SOPln (ar); → 4k
    ar[0] = 90;
    ar[1] = 50;
    fun (ar);
    SOP (ar[1]); → 5
}
```



Ex 5

```
public static void main () {
    float y = 7.84f; ✓
    int [][] mat = new int [3] [4];
    SOPln (mat) → 4k
    SOPln (mat[1]); → 6k
    SOPln (mat[1][3]); → 0
}
```

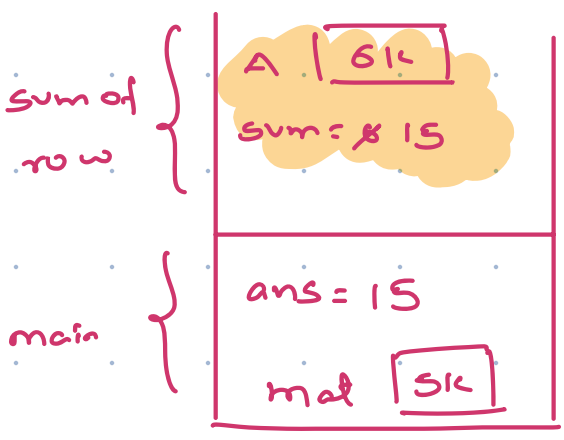


2D Array → Array of Arrays

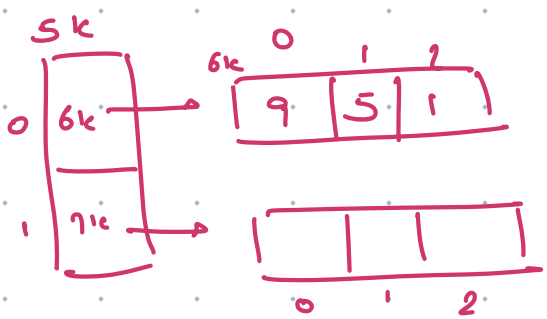
Ex 1

```
static int sumofrow(int[] A)        → 6k
{
    SOPln(A);  → 6k
    int sum = 0
    for(i=0; i<A.length; i++)
        sum = sum + A[i];
    return sum;
}
public static void main(){
    int[][] mat = new int[2][3];
    mat[0][0] = 9;  ←
    mat[0][1] = 5;  ←
    mat[0][2] = 1;  ←
    int ans = sumOfRow(mat[0]);
    SOPln(ans);  → 15
}
```

Stack



Heap



8:27 AM → 8:35 AM

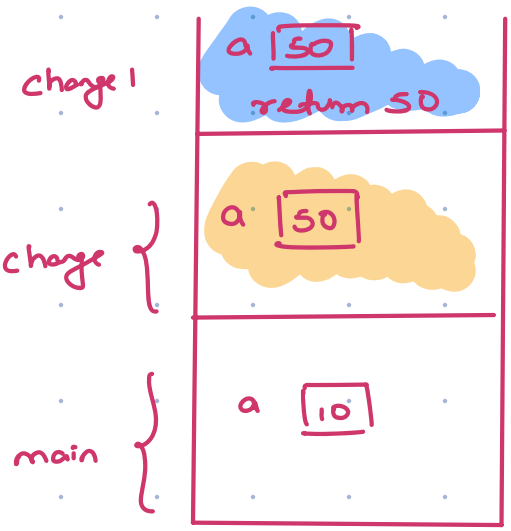
Predict the Output :

```
static void change(int a) {
    a = 50;
}

public static void main(String args[]) {
    int a = 10;
    change(a);
    System.out.println(a);
}
```

SOP(change1(a)); → 50

50

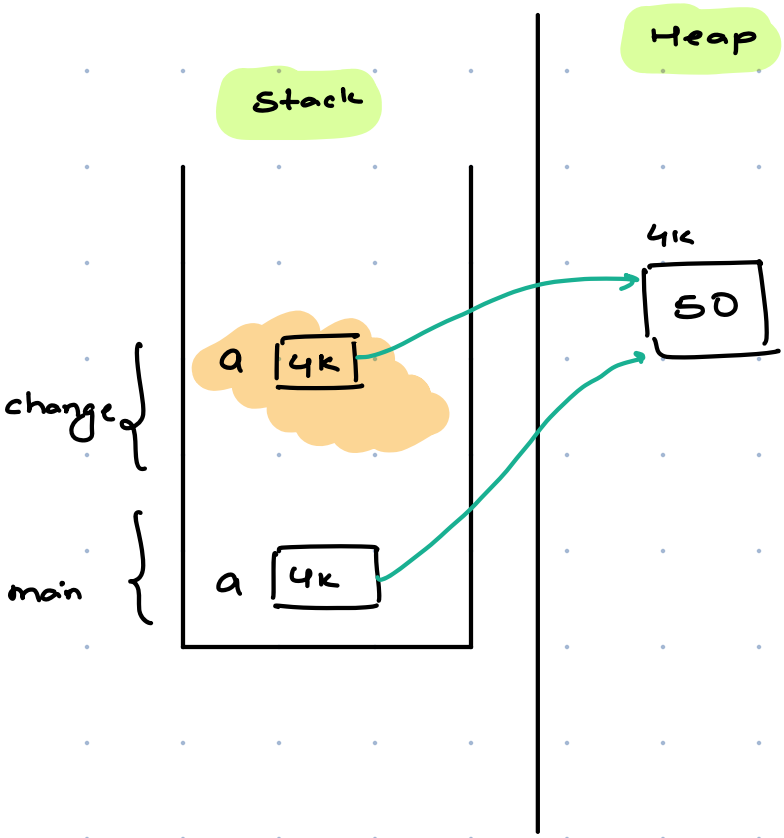
```
static int change1(a){
    a = 50
    return a;
}
```



Predict the output :

```
static void change(int[]a) {
    a[0] = 50;
}

public static void main(String args[]) {
    int[]a = {10};
    change(a);
    System.out.println(a[0]);
}
```

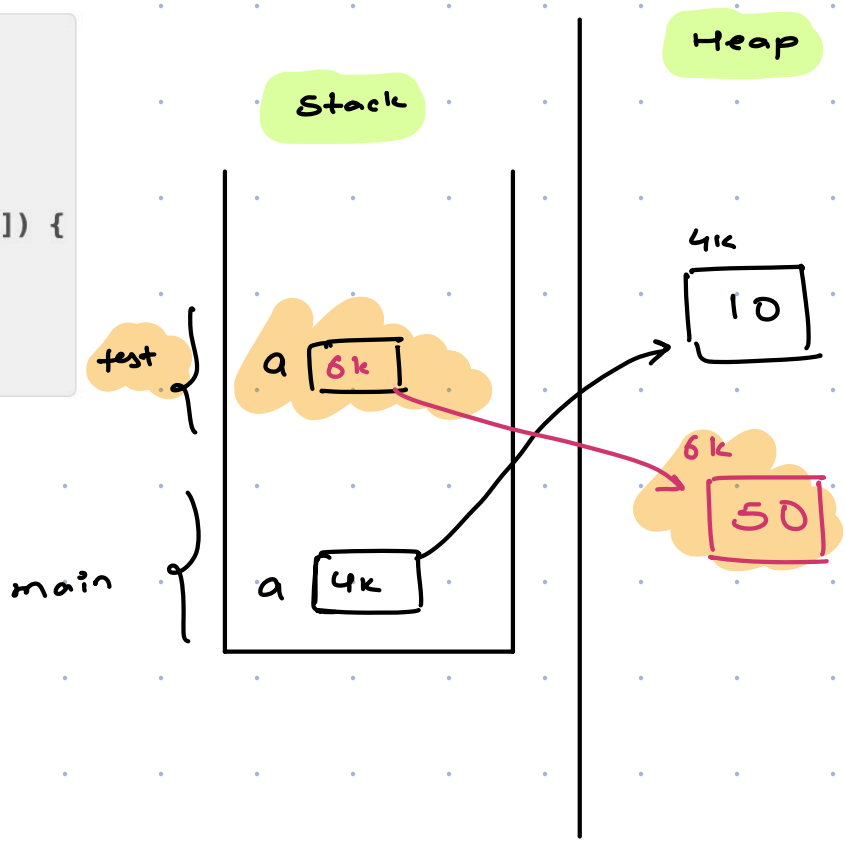
→ 50



Predict the output :

```
static void test(int[]a) {
    a = new int[1];
    a[0] = 50;
}

public static void main(String args[]) {
    int[]a = {10};
    test(a);
    System.out.println(a[0]);
}
```

→ 10

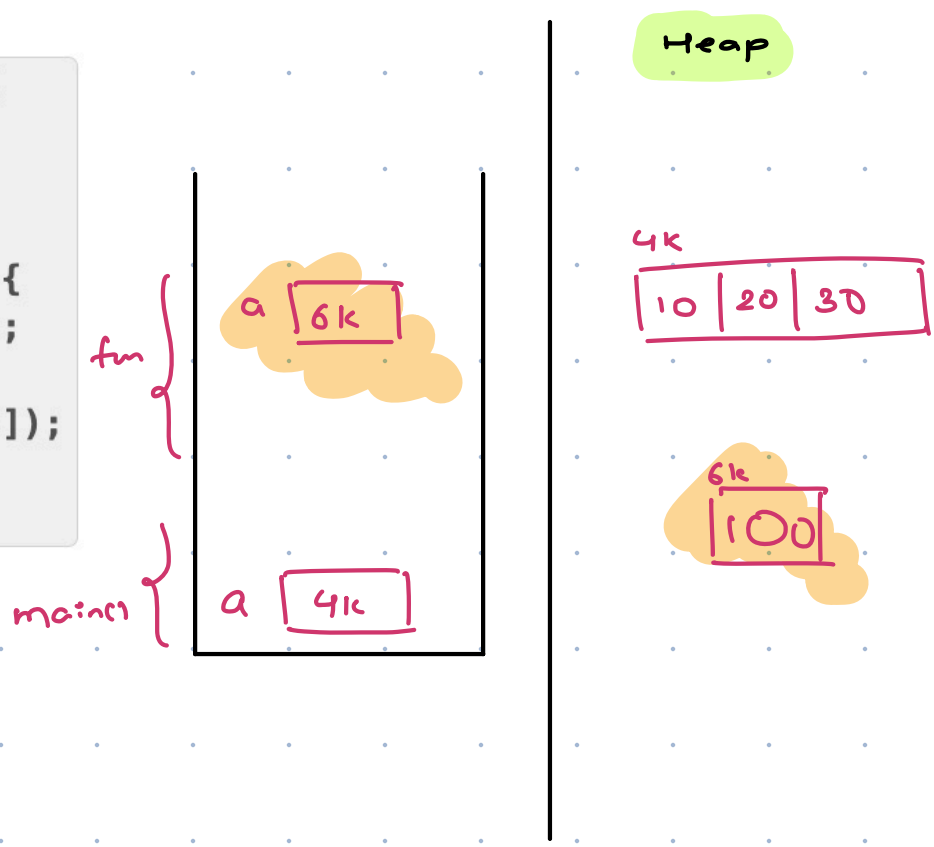


Predict the output:

```
static void fun(int[] a) {
    a = new int[1];
    a[0] = 100;
}
public static void main() {
    int[] a = {10, 20, 30};
    fun(a);
    System.out.println(a[0]);
}
```

↓↓

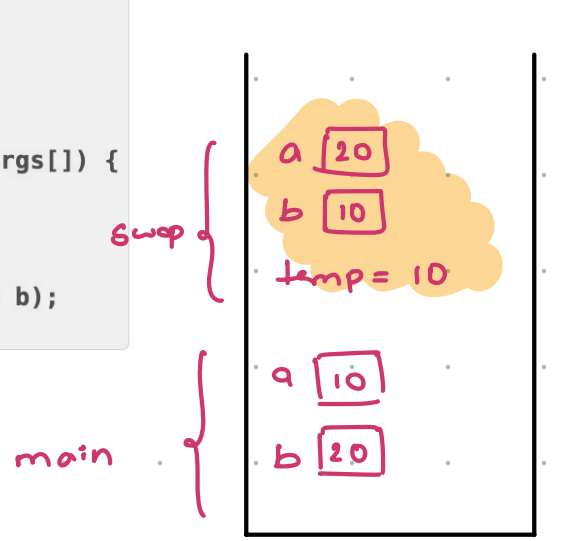
10



Predict the output :

```
static void swap(int a,int b) {
    int temp = a;
    a = b;
    b = temp;
}

public static void main(String args[]) {
    int a = 10;
    int b = 20;
    swap(a,b);
    System.out.println(a + " " + b);
}
```

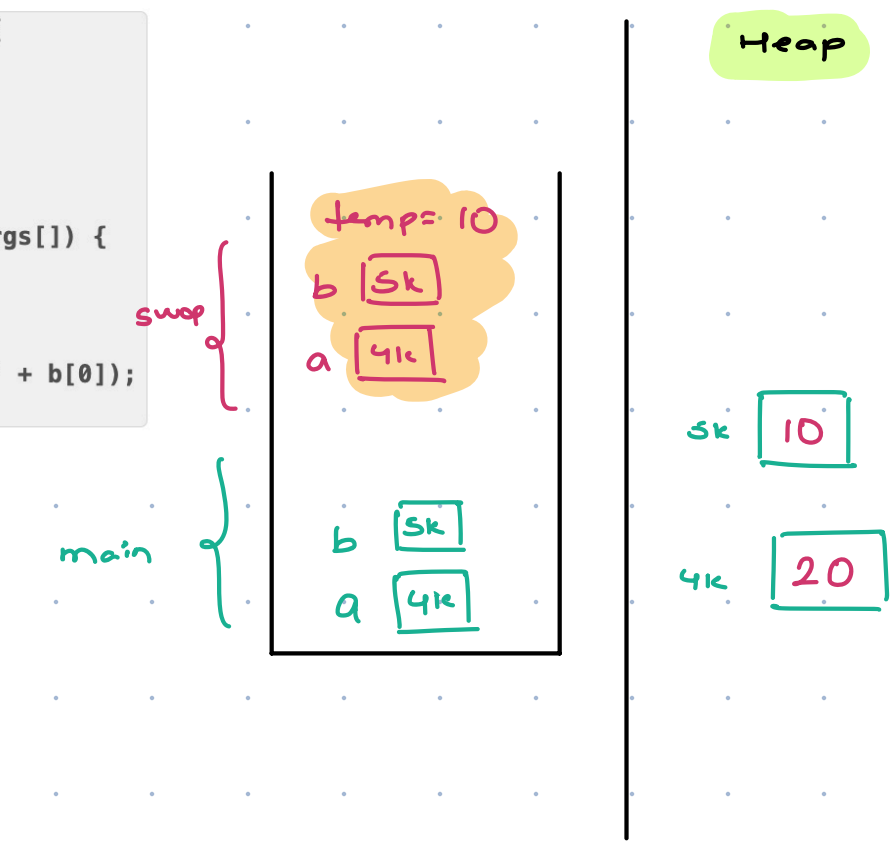
10 20



Predict the output :

```
static void swap(int[]a,int[]b) {
    int temp = a[0];
    a[0] = b[0];
    b[0] = temp;
}

public static void main(String args[]) {
    int[]a = {10};
    int[]b = {20};
    swap(a,b);
    System.out.println(a[0] + " " + b[0]);
}
```

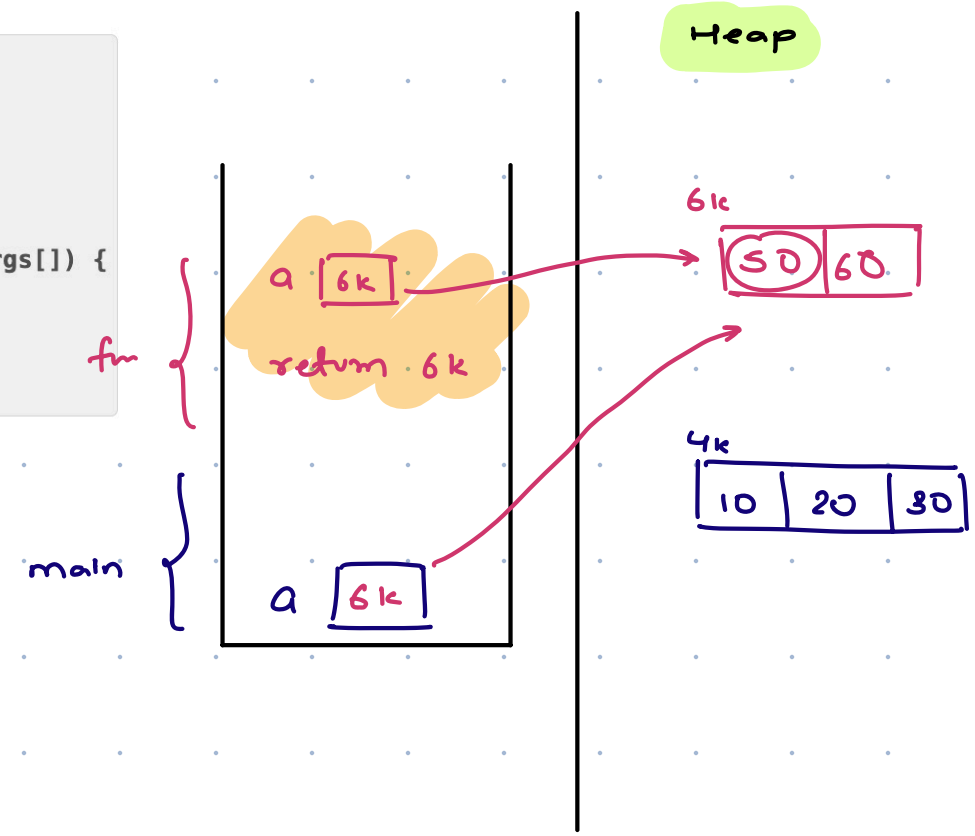
20 10



Predict the output :

```
static int[] fun(int[]a) {
    a = new int[2];
    a[0] = 50; a[1] = 60;
    return a;
}

public static void main(String args[]) {
    int[]a = {10,20,30};
    a = fun(a);
    System.out.println(a[0]);
}
```

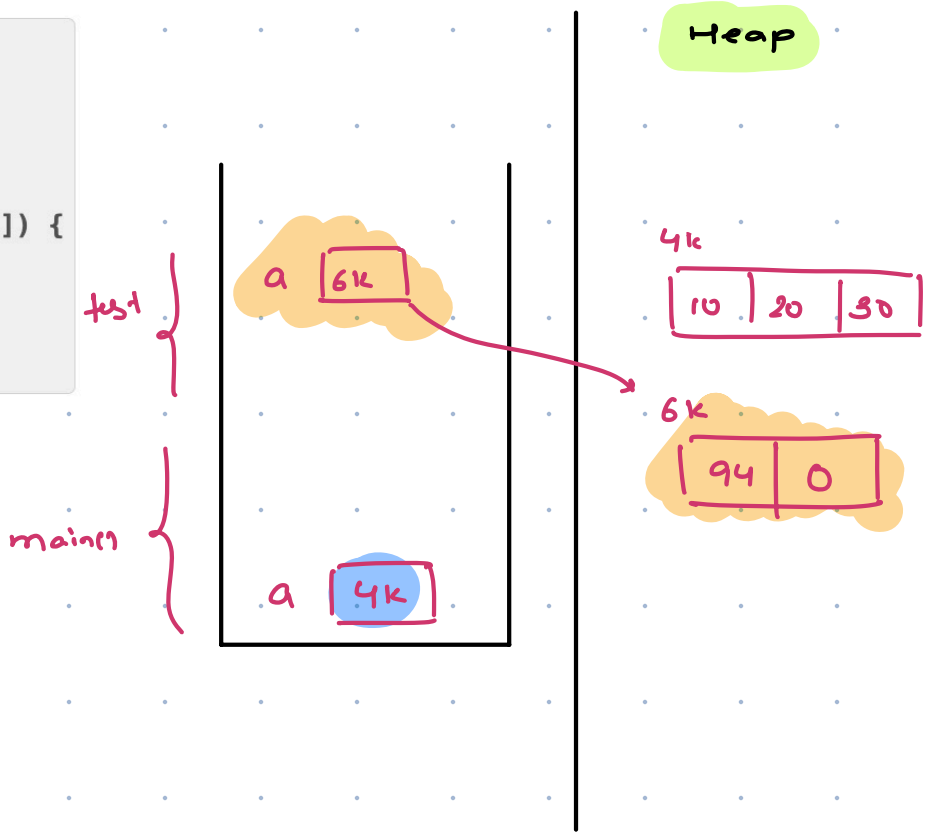
50



Predict the output :

```
static void test(int[]a) {
    a = new int[2];
    a[0] = 94;
}

public static void main(String args[]) {
    int[]a = {10,20,30};
    test(a);
    System.out.println(a[0]);
}
```

4k[0] ⇒ 10



Doubts

Address of Array = Value of A

int [] A = ____

String S = Address of arr

if (Address of Arr == S) {

}

Prefix sum / Cf Approach / Subarrays &

Sliding window

Antidiagonals

AL<AL<I> ans = new AL<>();

```
AL<I> smallas = new AL<>();

{
    ——
    ——
    ——
    ——

    smallas.add(A[i][j]);
}

ans.add(new AL<>(smallas));
```

Pick B elements

A[] = { 2 3 4 5 6 7 8 }

scenario — i ↑ — j ↑

1' 4 0 = 14 + 0 = 14

(2) [3] 1 = 14 − 5 + 8 = (17)

3 [2] 2 = 17 − 4 + 7 = 21

4 [1] 3 = 21 − 3 + 6 = 24

5 [0] 4 = 24 − 2 + 5 = 27