Parikshit 2019 IIITD

SDE3 @ Amazon

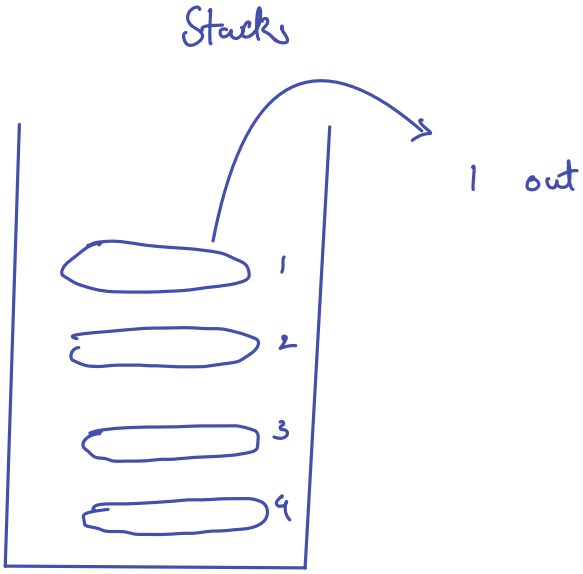
2.5 years scaler

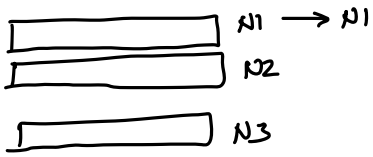
Memory Management

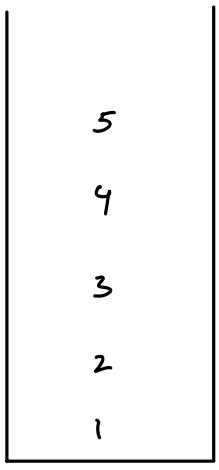
RAM

Agenda

- How program works
- Quizes







single opening ds

1) Insert ⟶ push
2) remove ⟶ pop

push (1)
push (2)
push (3)
push (4)
push (5)
pop() ⟶ 5

LIFO
{Last In First Out}

RAM

Function call stack

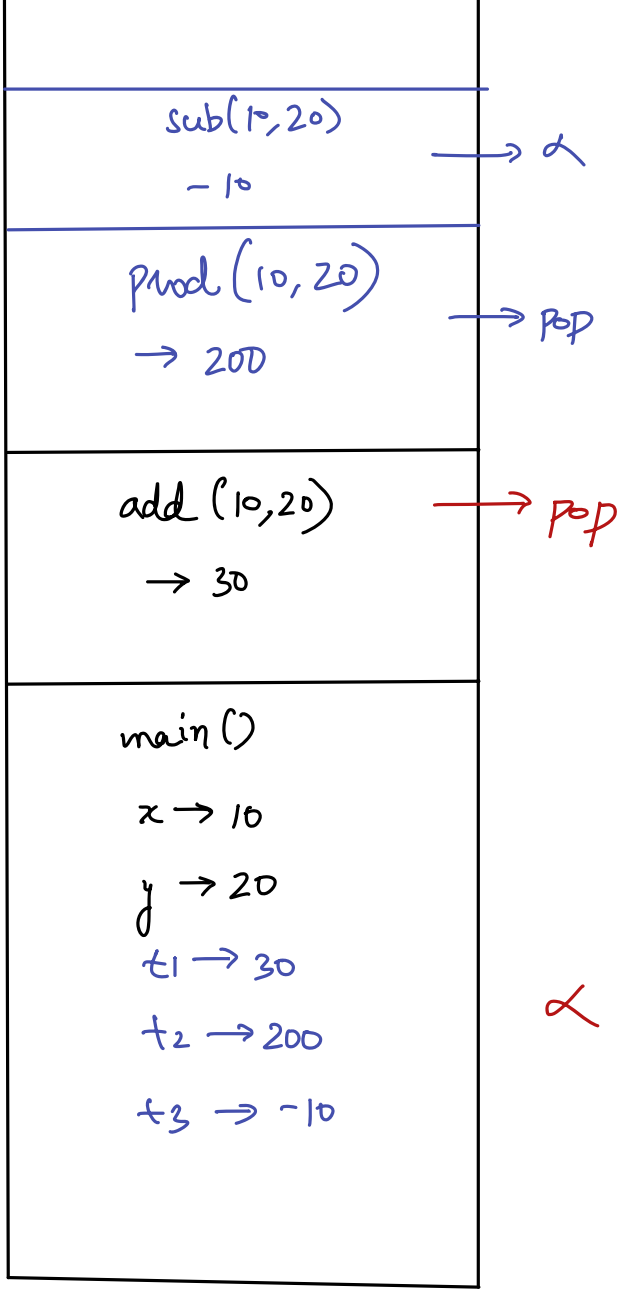
print ∝

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

```
int prod (x, y) {
    return x*y
}
```

```
int sub(x, y) {
    return x-y
}
```

```
void main() {
1   x = 10
2   y = 20
3   t1 = add(x, y)
4   t2 = prod (x, y)
5   t3 = sub(x, y)
6   print (t3)
}
```

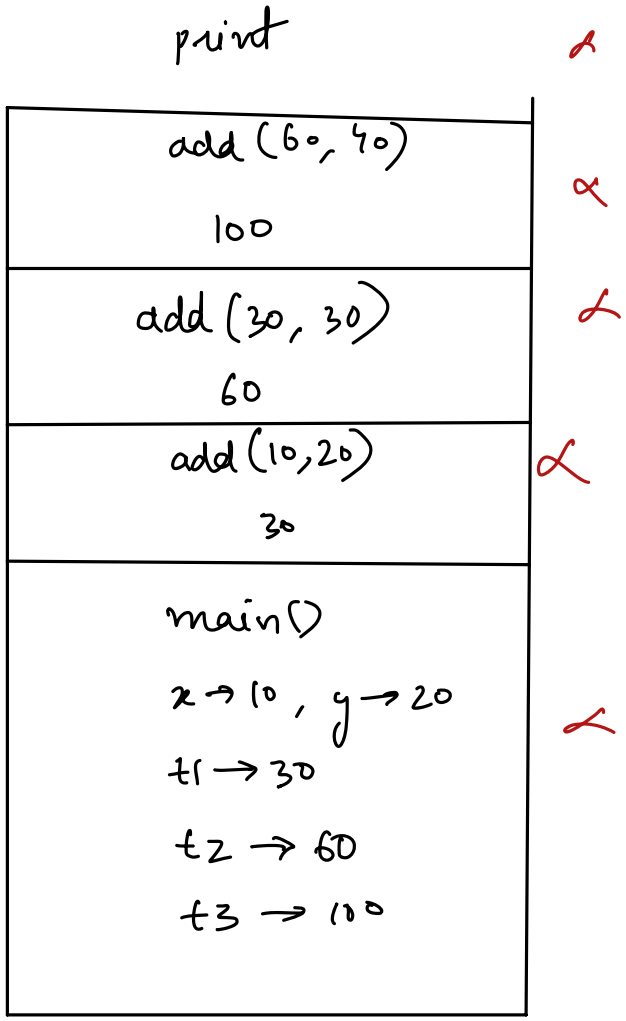


execute → push

fall back → pop

console : -10

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

void main(){
    x = 10
    y = 20
    t1 = add (x, y)
    t2 = add (t1, 30)
    t3 = add (t2, 40)
    print (t3)
}
```



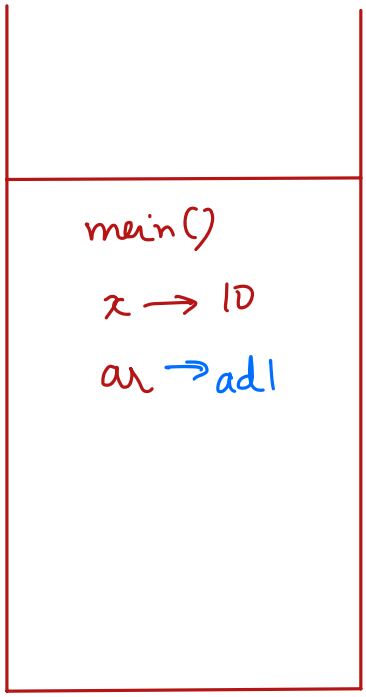
console : 100

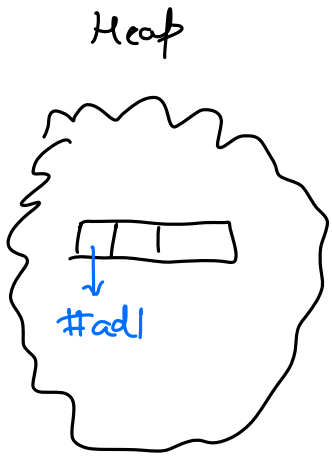
premitive data types are always pass by value

Types of memory

stack	heap
small	large
primitive data types int, char, bool, long, float	non primitive data arrays, strings, objects

```
void main() {
    x = 10
    ar = new int [3]
    print (ar)
    print (ar[2])
    ar[2] = 2
}
```

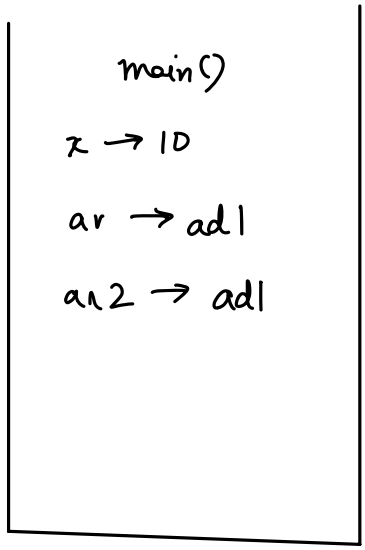


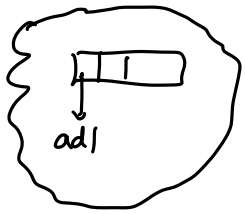


RAM
→ stack {smaller}
→ heap {larger}

Q

```
void main() {
    x = 10
    ar = new int [3]
    ar2 = ar
    print (ar) → adl
    print (ar2) → ad
}
```



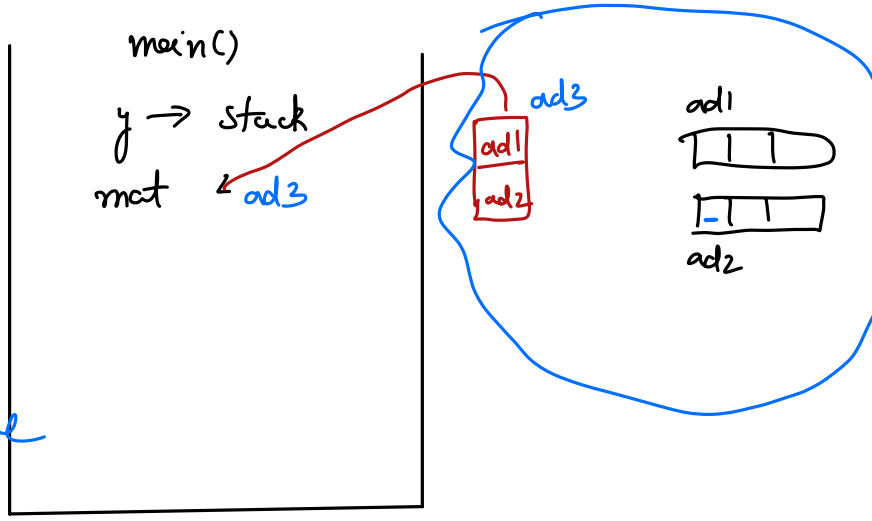


adl

adl

Q

```
void main() {
    y = 7.84
    mat = new int[2][3]
    print (mat) → add3
    print (mat[1]) → add2
    print (mat[1][0]) → value
}
```



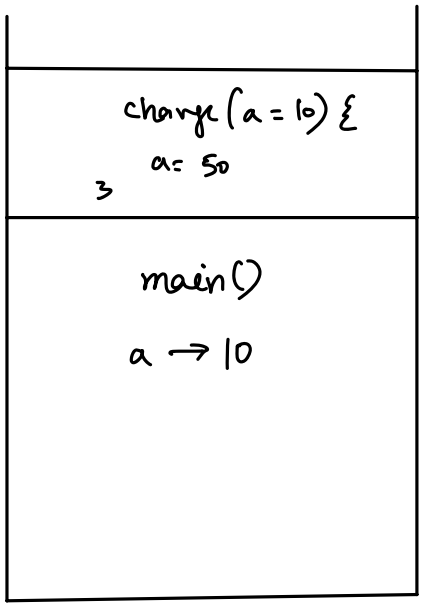
Break (10:00 - 10:10)

Q

```
void sum ( mat[][]) {
    print (mat)
    print (mat[0][1] + mat[1][0])
}
```

Quizes

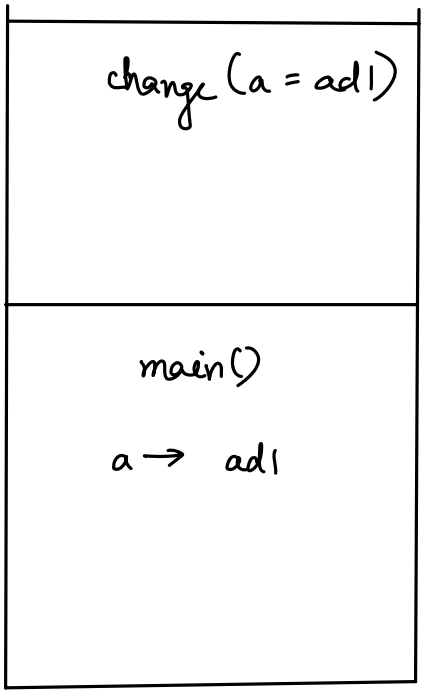
```
void change (a) {
    a = 50
}

void main () {
    a = 10
    change (a)
    print (a) → 10
}
```

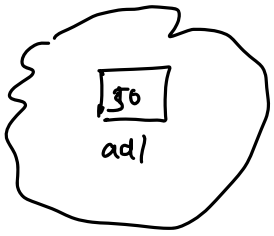


pass by value

```
void change (a[]) {
|       a[0] = 50
}
```

```
void main () {
|   a[] = {10}
|   change (a)
|   print (a[0]) → 50
}
```





pass by reference

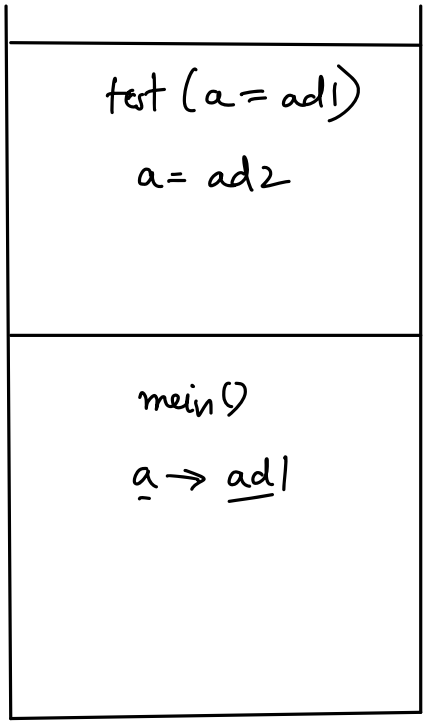
a = 3

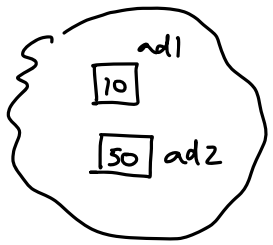
a = 5

print(a) #5

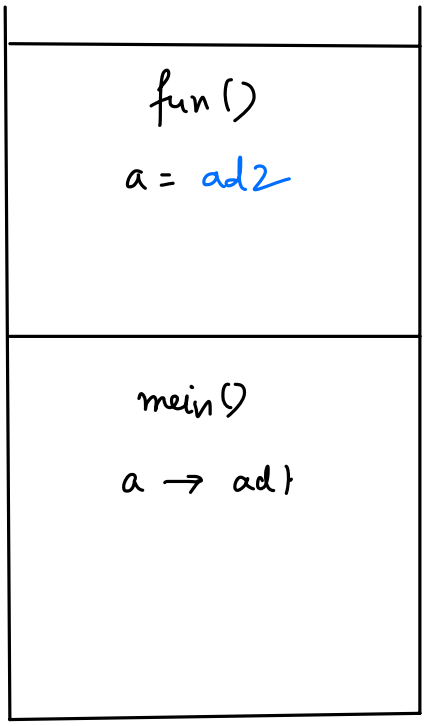
```
void test (a[]){
    a = new int [1]
    a[0] = 50
}
```

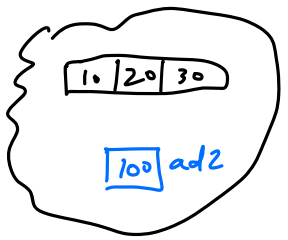
```
void main () {
    a[] = {10}
    test (a)
    print (a[0]) → 10
}
```



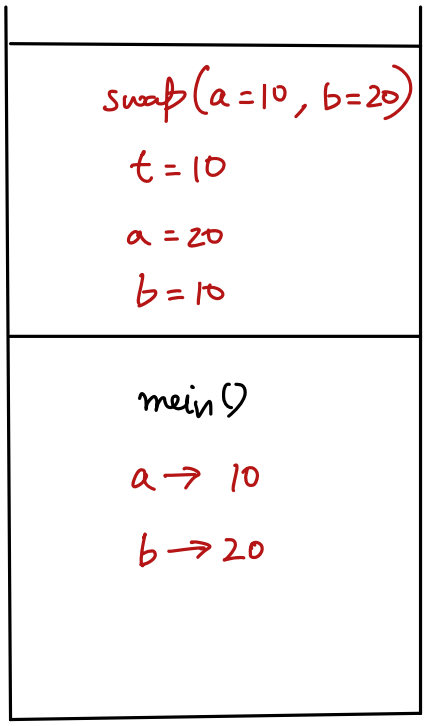


```
void fun(a[]){
    a = new int[1]
    a[0] = 100
}

void main(){
    int a[] = {10, 20, 30}
    fun(a)
    print(a[0])        # 10
}
```



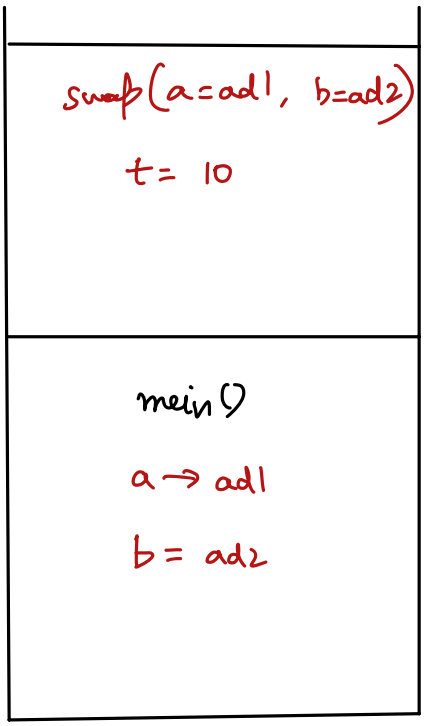


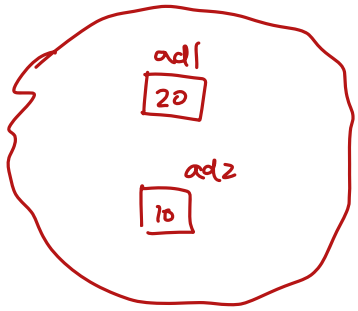
```
void swap(a, b) {
    t = a
    a = b
    b = t
}

void main() {
    a = 10
    b = 20
    swap(a, b)
    print(a + " " + b)   10  20
}
```



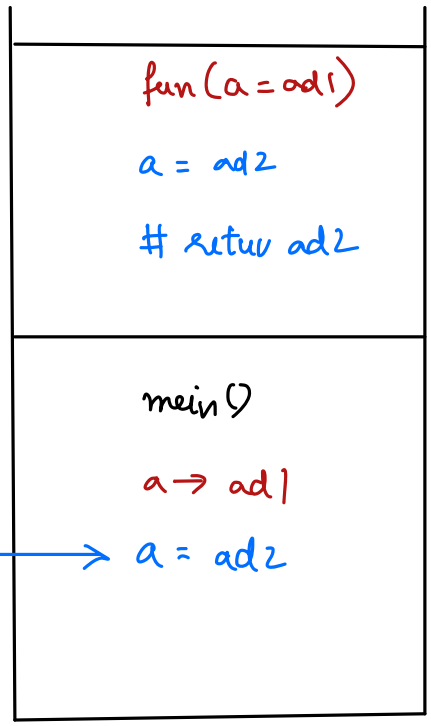
```
void swap (a[], b[]) {
    t = a[0]
    a[0] = b[0]
    b[0] = t
}

void main() {
    a[] = {10}
    b[] = {20}
    swap (a, b)
    print (a[0] + " " + b[0])
}
```

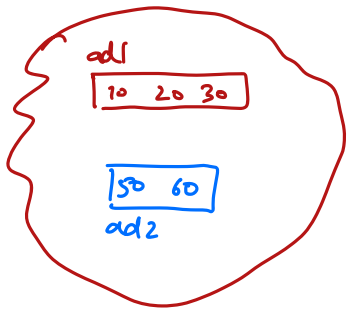
20 10





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

void main(){
    a = {10, 20, 30}
    a = fun(a)
    print(a[0])   # 50
}
```





a[3]