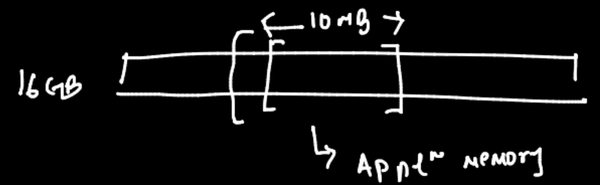
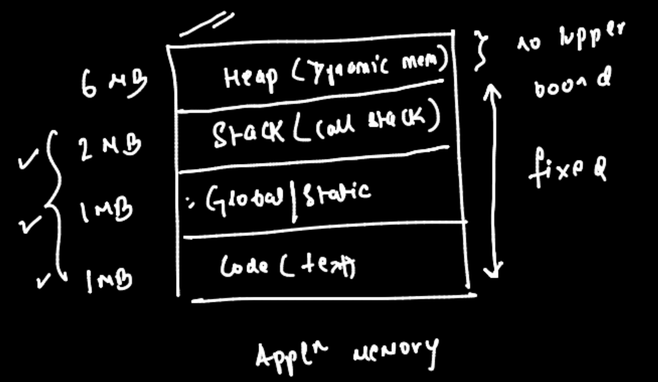
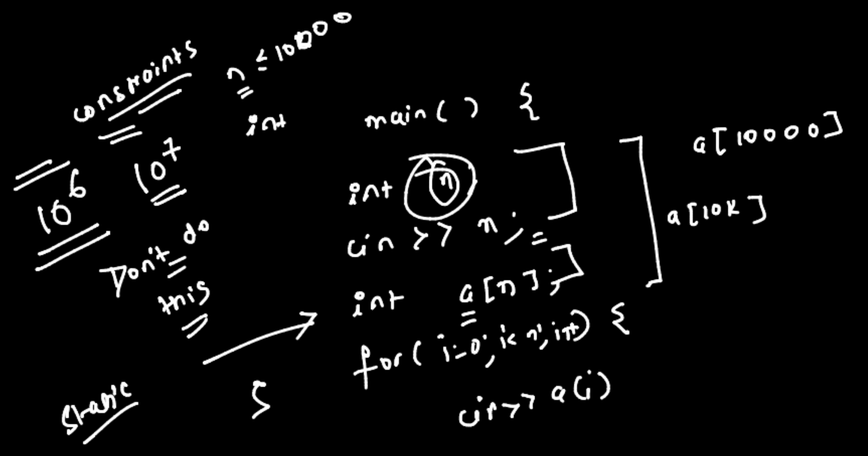






Stack

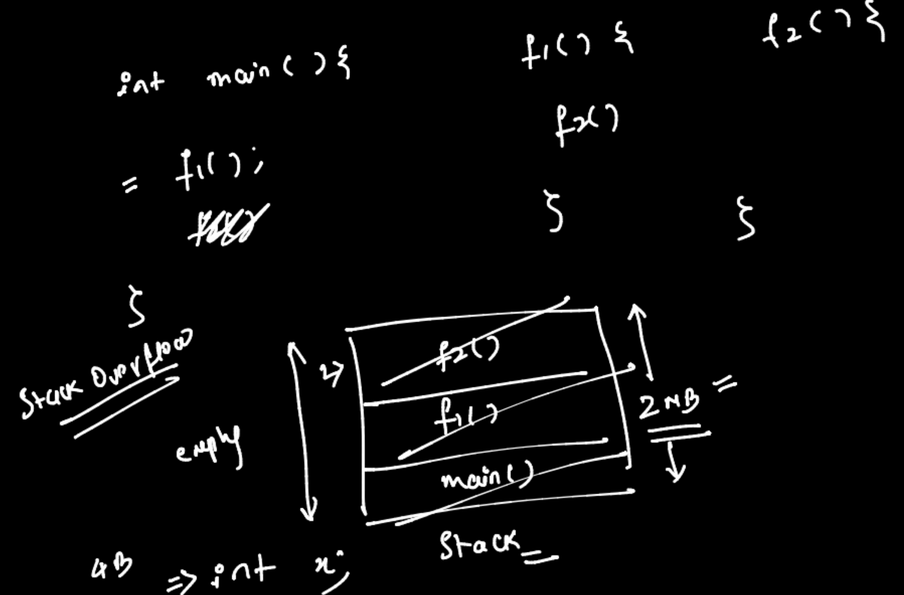
- when you compile a program the space for stack-frame is fixed - static allocation

4B

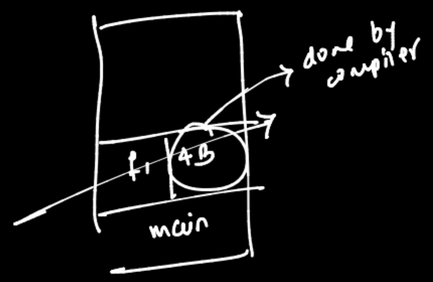
```
f1() {
  int a;
}
```

8B

```
f2() {
  double d;
}
```



```
f1() {
    int x;
}

main() {
    f1()
}
```



all + deall

1. Stack memory is fixed
2. All + Deall is done by compiler
3. Array → specify the size at compile time

OS, COA, Compile Design

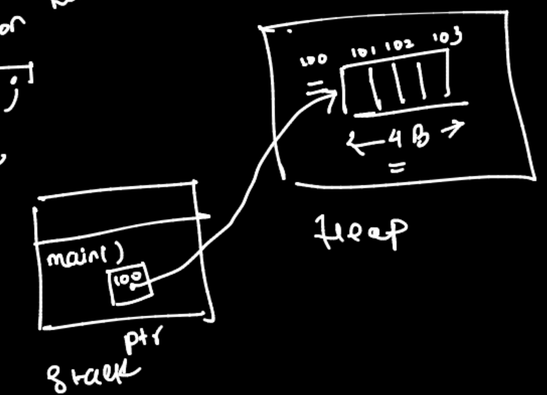
Main ✗ System

Dynamic Memory / Heap

"new" Keyword

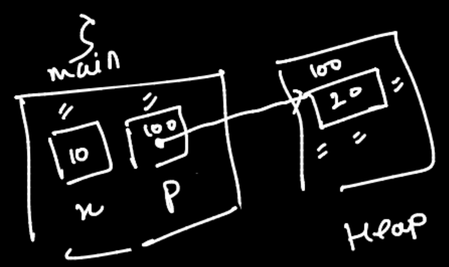
Reserve 4B of space on Heap

```
int main(){
    int * ptr = new int;
}
```



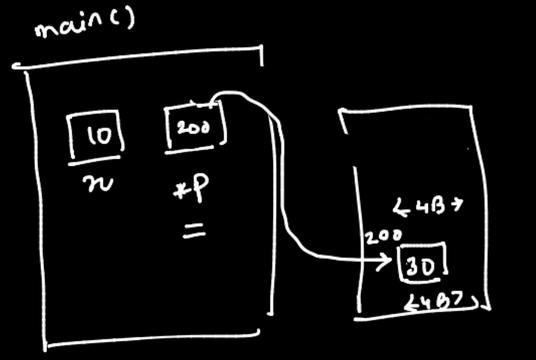

```
int main(){
    int x=10;
    int *p;
    p = new int;
    return 0;
}
```

```
*p = 20;
cout << *p; // 20
```



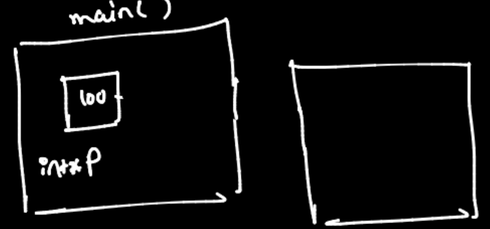

```
new char;   // 1B
new double; // 8B
new float;  // 4B
```

```
int main() {
    int x = 10;
    int * p = new int;
    *p = 20;
    cout << *p << endl;
⇒   delete p;
⇒   p = new int;
    cout << *p << endl;
    *p = 30
    cout << *p; // 30
```



⇒ memory leak

```
int main() {
    int * p = new int;
    *p = 20;
    cout << *p;
⇒   delete p;
⇒   p = NULL;
    cout << *p;
```

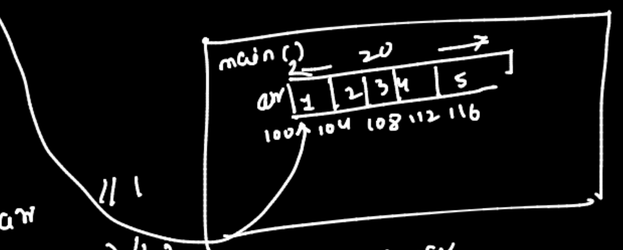


= * (addr 100) // garbage

O/P : 20

```
main() {
    int arr[5];
}
```

size should be given at the compile time



arr[0] or *arr // 1

arr[1] or *(arr+1) // 2

arr[2] or *(arr+2) // 3

arr[i] or *(arr+i)

Stack

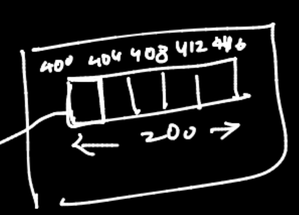
Dynamically (create on integer ⇒

```
int * ptr = new int[size];
```

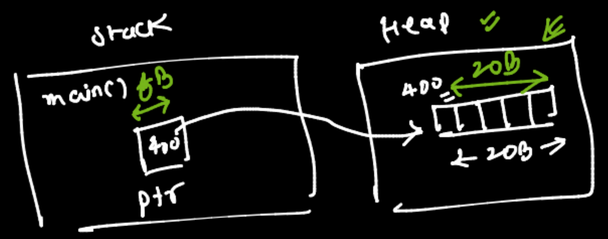
= compile time =
or
= runtime (✓)

```
int * ptr = new int[5];
```

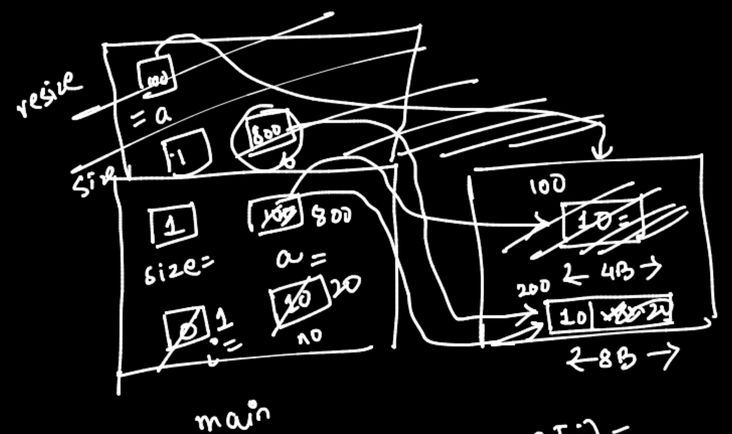
↳ 5*4 = 20B



```
main() {
    int * ptr = new int[5];
}
```



*ptr ⇒ ptr[0] =
*(ptr+1) ⇒ ptr[1] =
*(ptr+2) ⇒ ptr[2] =
*(ptr+i) ⇒ ptr[i] =



ptr a[] or int* a
f() {

}

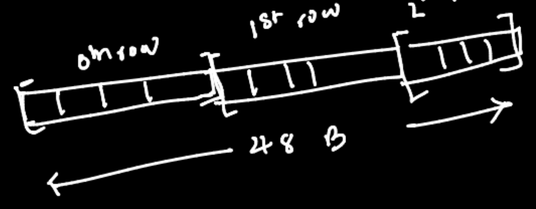
```
main() {
    int a[] = {1, 2, 3};
    f(a);
}
```

2D Allocation

```
main() {
    int a[3][4];
}
```

⇒ 3 1D arrays of size 4
12 × 4 = 48 B

⇓
row-major



I want to create

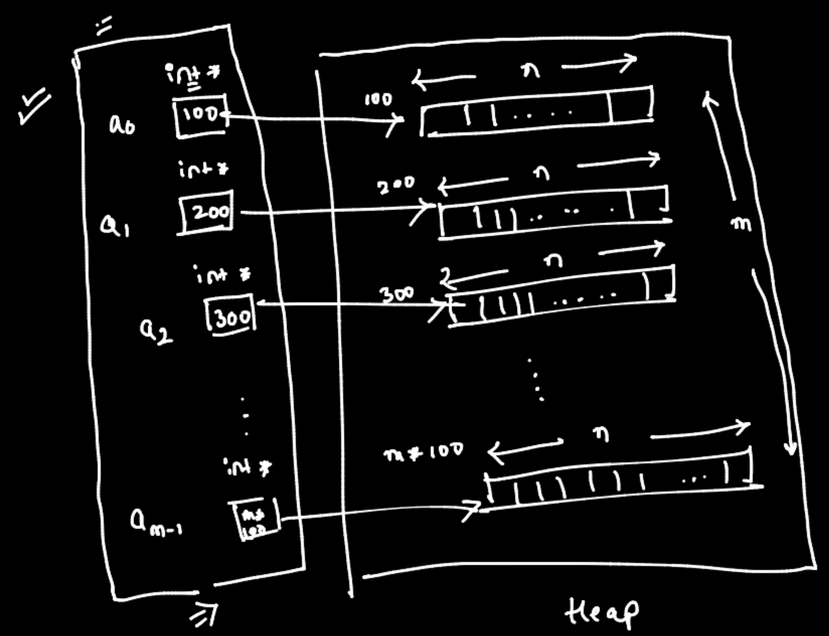
a 2D array (m x n) dynamically

⇓

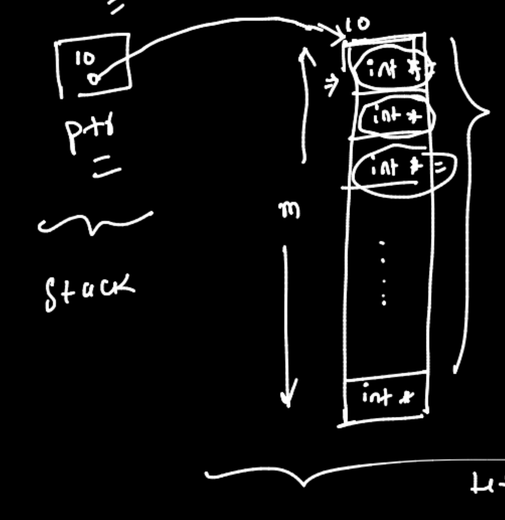
'm' 1D arrays of size 'n' dyna.

m 1D arrays of n each

```
int* a0 = new int[n];
int* a1 = new int[n];
int* a2 = new int[n];
        :
int* am-1 = new int[n];
```




```
int **ptr = new int*[m];
```



ptr[0] = *(ptr+0)

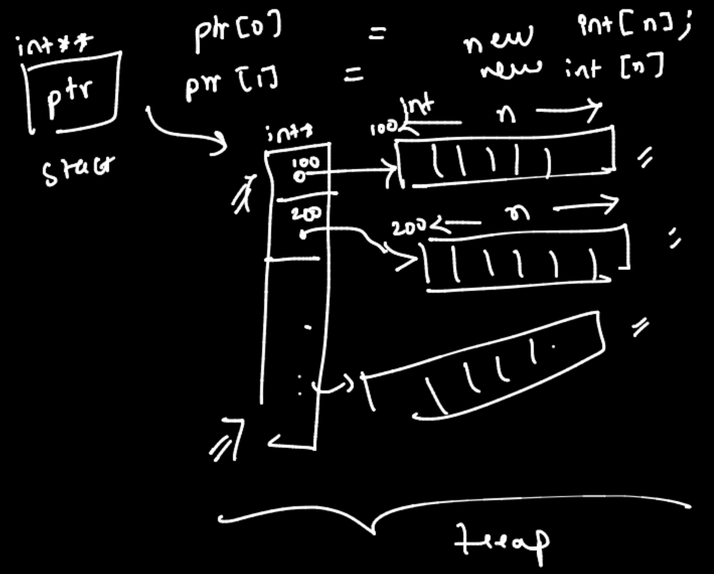
ptr[1] = *(ptr+1)

ptr[2] = *(ptr+2)

⋮

ptr[i] = *(ptr+i)

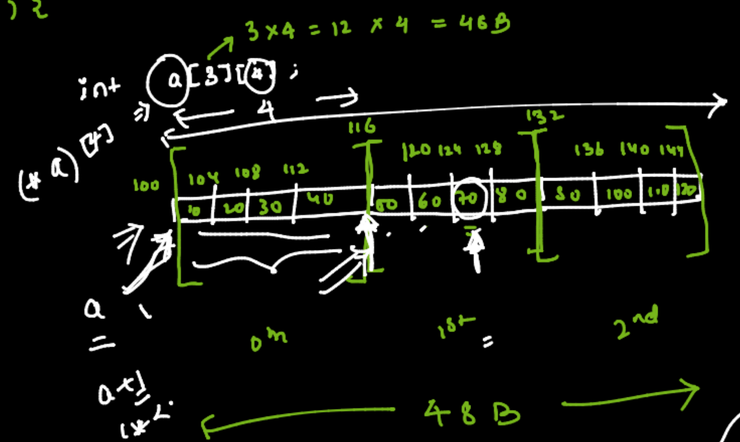
```
int **ptr
ptr[0] = new int[n];
ptr[1] = new int[n]
```



f() {

int (*a)[3] ⇒ a[3][4];

3×4 = 12 × 4 = 48B



}

a[1] ⇒ 48B → width

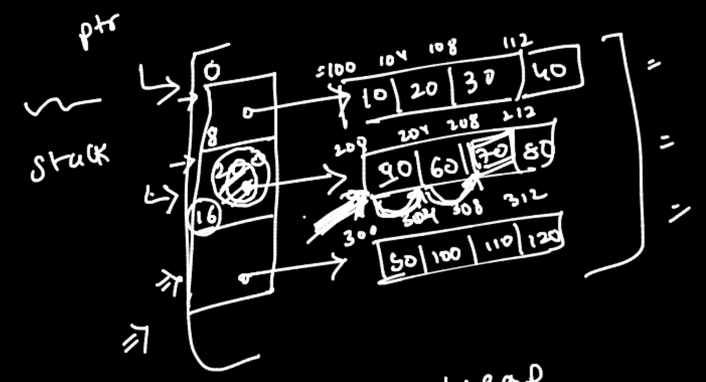
a[1][2] ⇒ 70 ⇒ a + 1 * <4> * 4 + 2 * 4

Single dereference

100 + 16 = 116 + (2*4)

= 116 + 8

= *124 = 70



out << ptr[1][2] ⇒ 70

= * (*(ptr + 1) + 2) ⇒ 70

⇒ 2 dereference

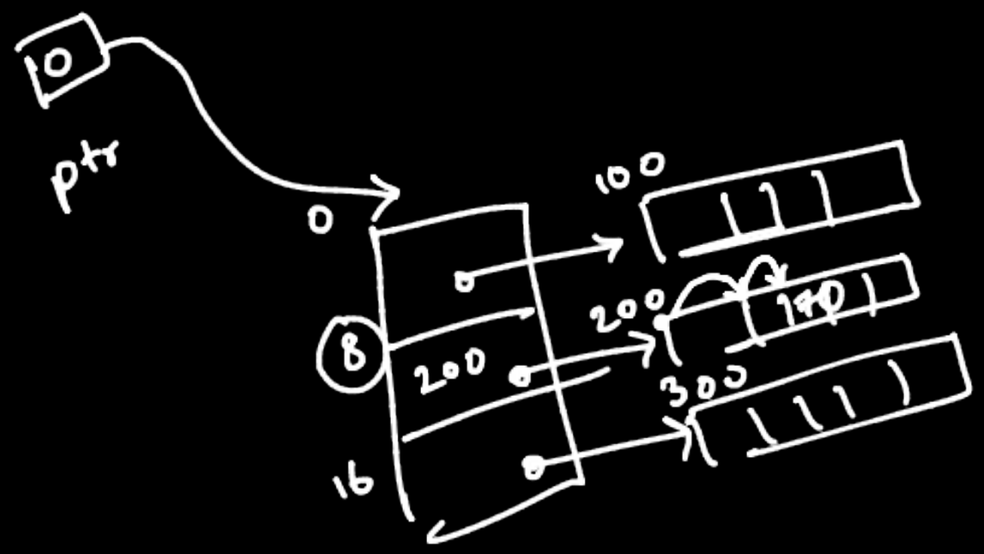
```
int ** ptr = new int*[3];
for(int i=0; i<3; i++) {
    ptr[i] = new int[4];
}

for(int i=0; i<3; i++) {
    for(int j=0; j<4; j++) {
        cin >> ptr[i][j]
    }
}
```

You cannot use 2D array which statically & dynamically are interchangably.

f(int **a)

⇒ int a[3][4];

f(a)



ptr[1][2]

= *(ptr + 1)

= *(8)

= *(200 + 2)

= *(208) = 70