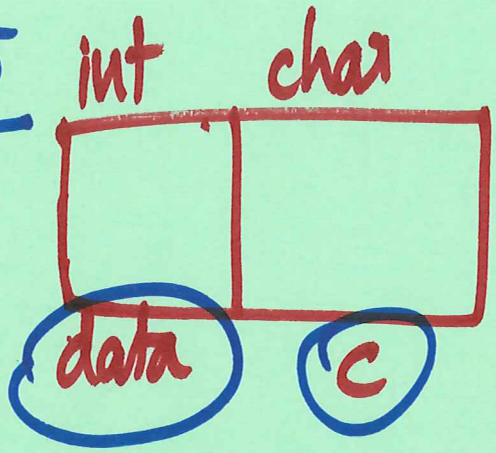
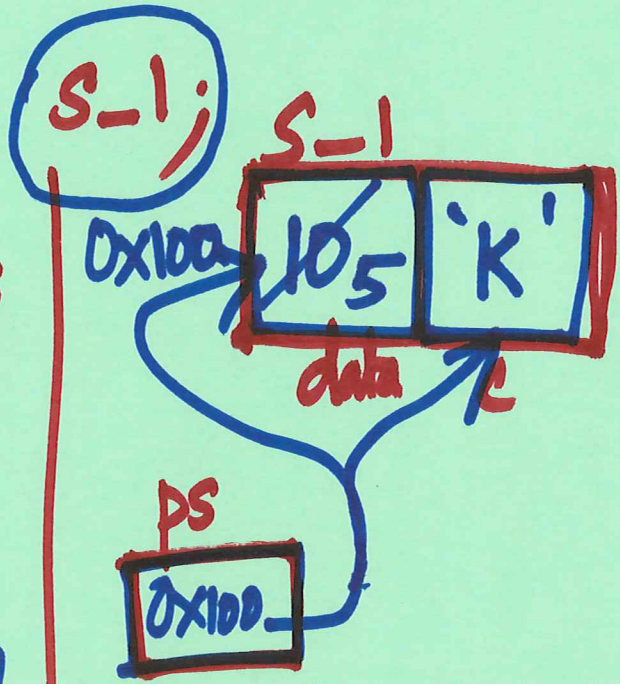
Lecture - 5

```
struct s
{
    int data;
    char c;
};
```




```
struct s s_1;
struct s *ps;
```

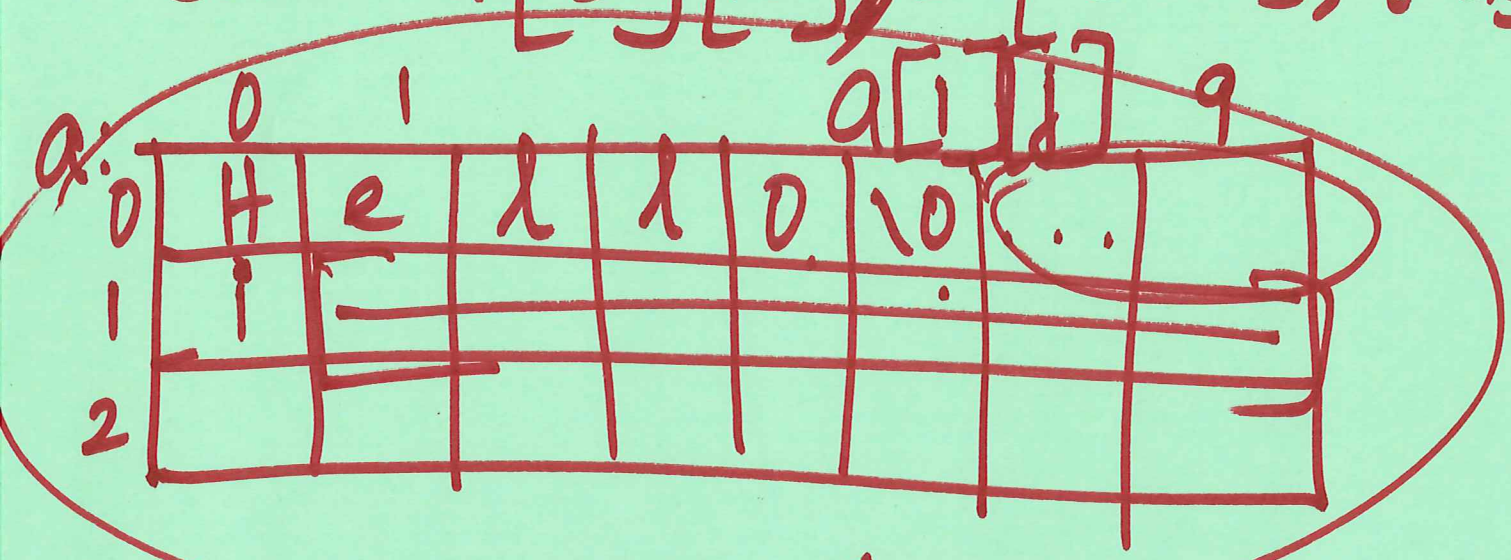



```
ps = &s_1;
```

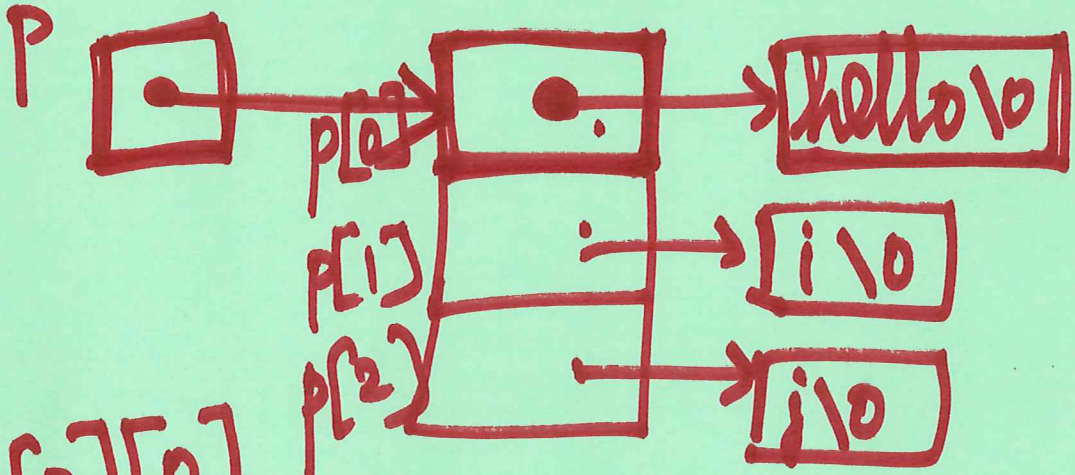
```
ps -> data = 10;
ps -> c    = 'K';
```

```
s_1.data = 5;
```

char a[3][10] = {{"Hello"}, {"i"}, {"j"}};



char *p[] = {"hello", "i", "j"};



*(p+0)

p[0]

p[0][0]

int a[10];

int a[10][20];

int *a;

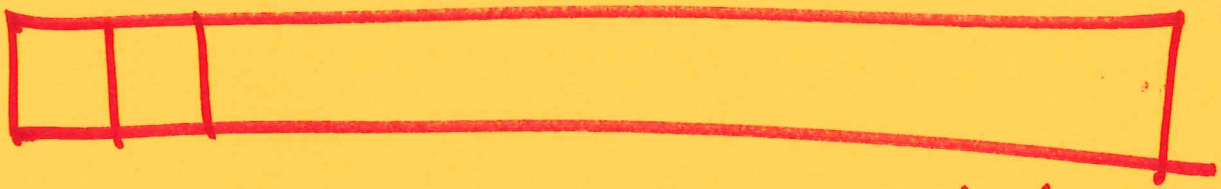
int *a[];

int a[10];

1). size fixed

2). insert at beg / middle

3). int a[1000]; 1000 x 4 = 4000 bytes



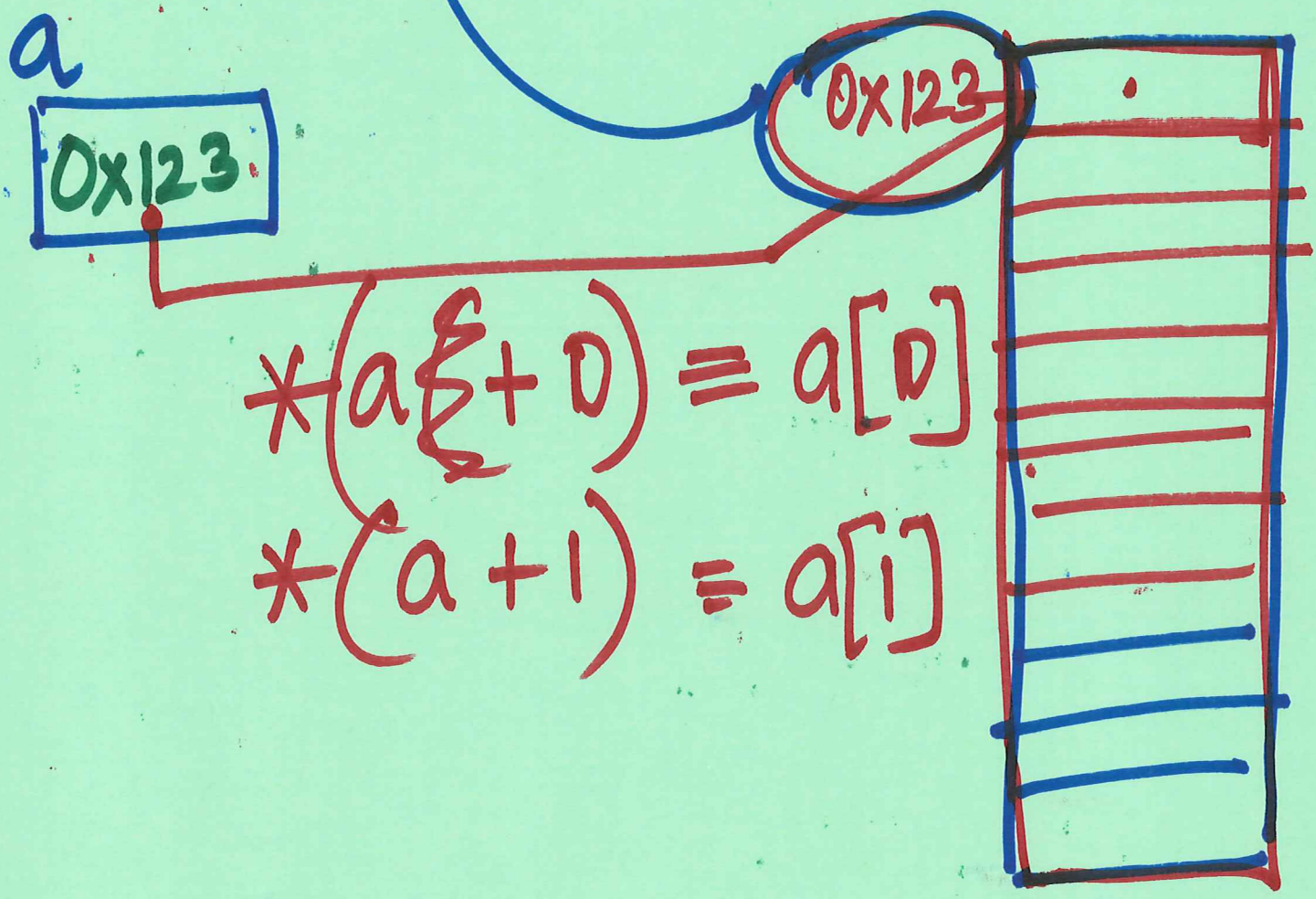
40 x 4 = 160 bytes

Dynamic Memory Allocation.

void * malloc (size of bytes)

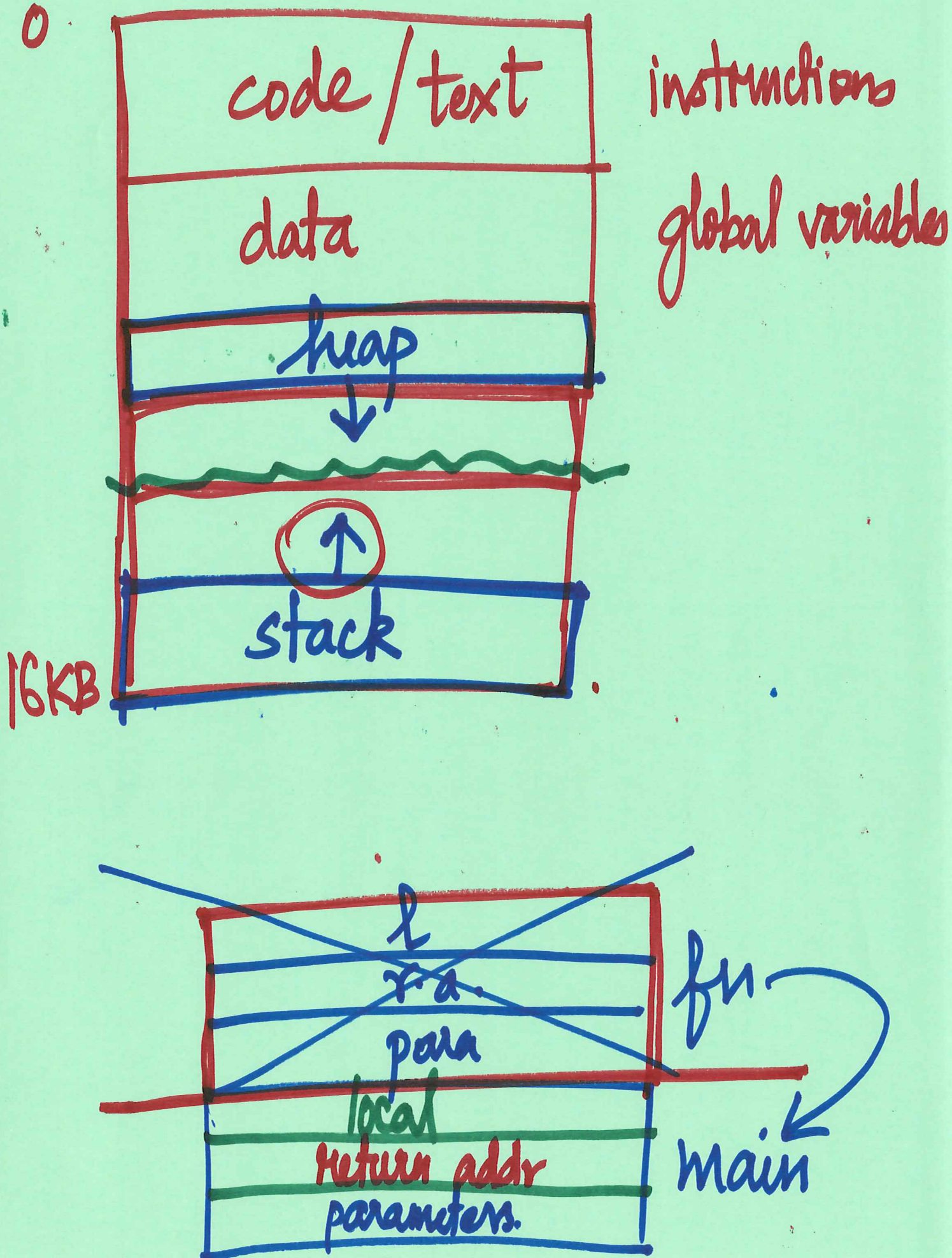
int *a = malloc (n * sizeof (int));

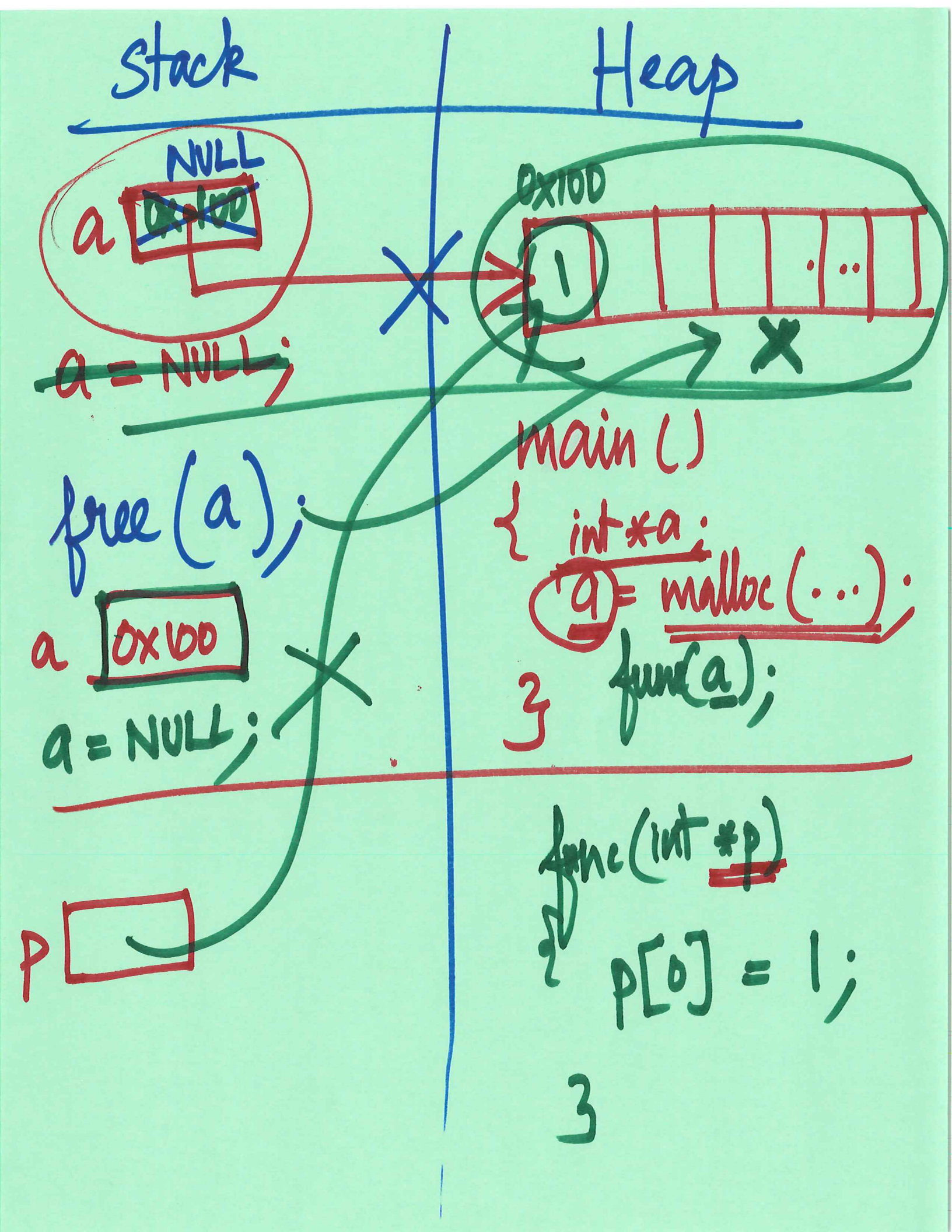
user



*(a + 0) ≡ a[0]

*(a + 1) ≡ a[1]


0
code / text
instructions
data
global variables
heap
stack
16KB
l
r
a
para
local
return addr
parameters
fn
main


Stack
Heap
NULL
a
0x100
0x100
1
a = NULL;
free(a);
main()
{ int *a;
a = malloc(...);
func(a);
}
a
0x100
a = NULL;
p
func(int *p)
{
p[0] = 1;
}

```
struct node
{
    int data;
    struct node *next;
};
```