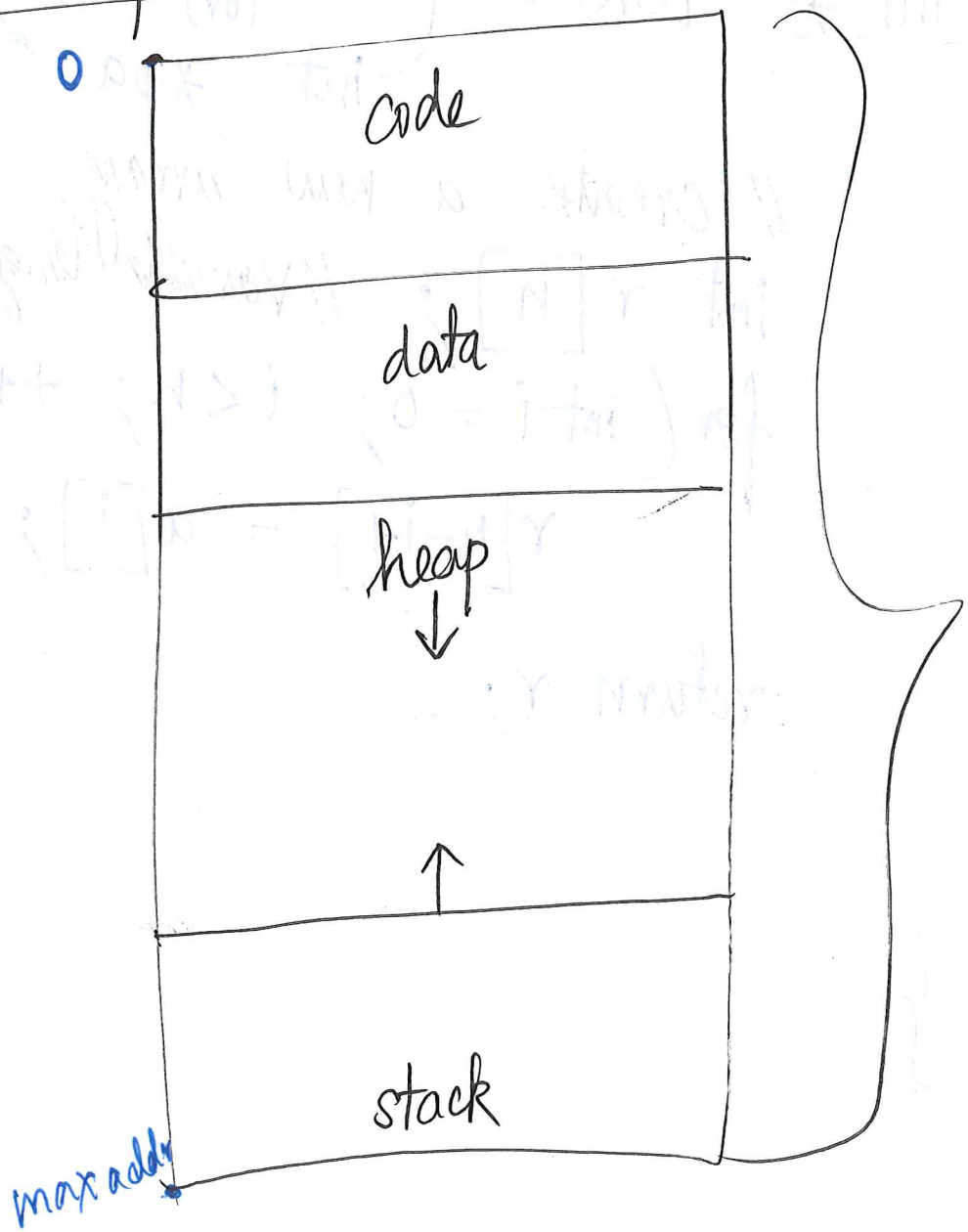
CS 354 - Lecture

Today

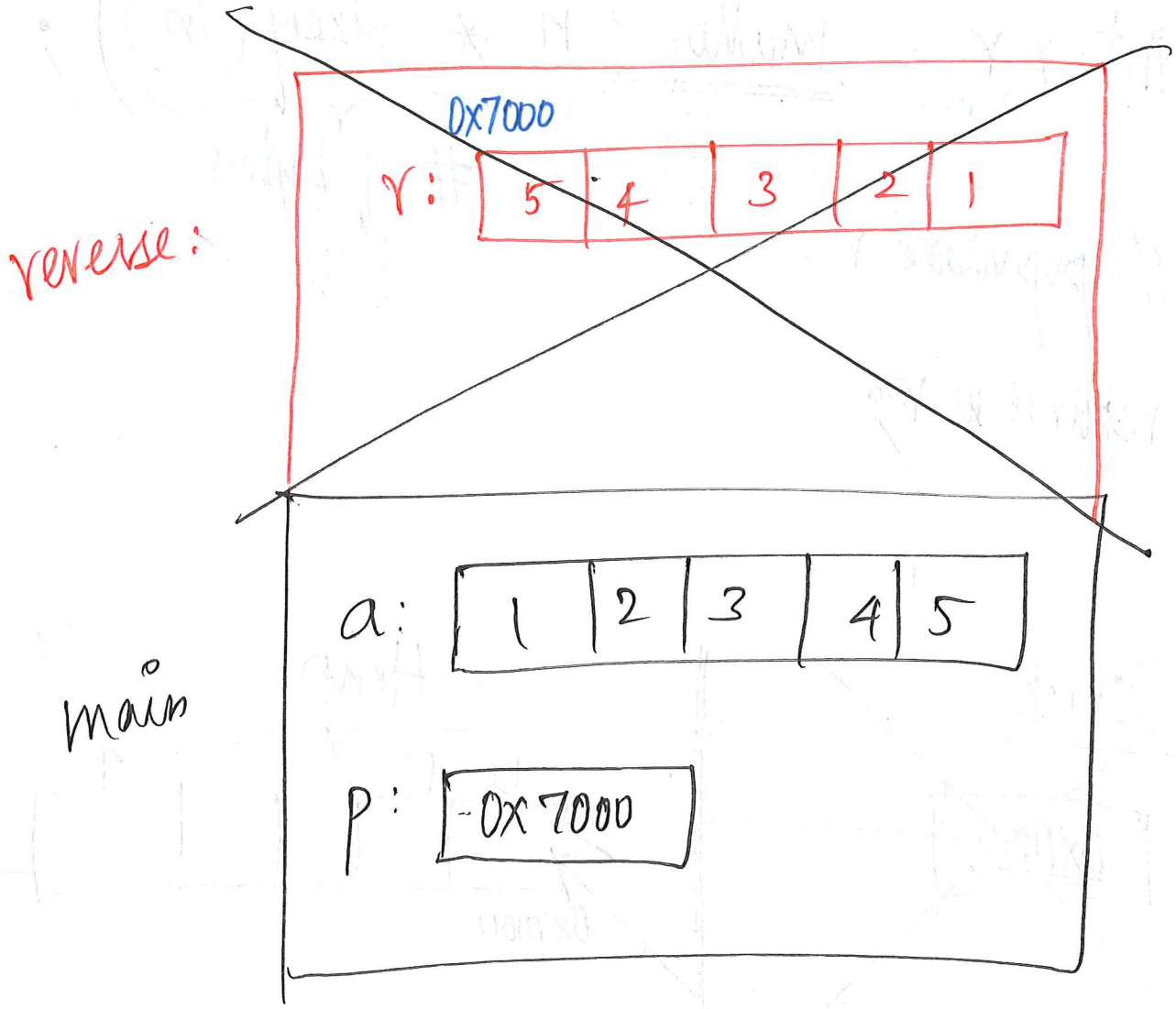
1. Dynamic memory allocation. (Heap)
2. Dynamic arrays.
3. Linked Lists

Address Space



(2)

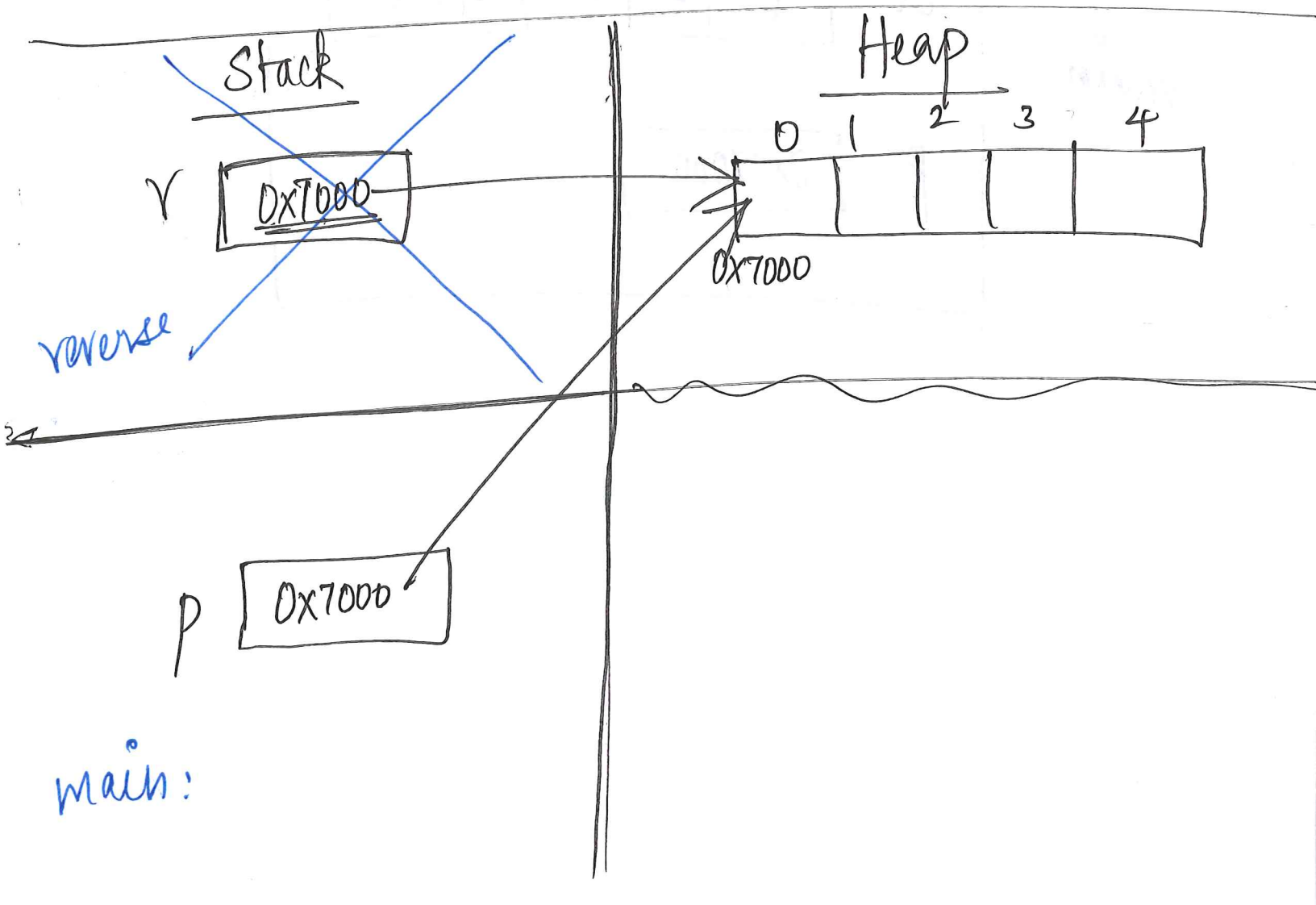
```
main() {
    int a[5] = {1, 2, 3, 4, 5};
    int *p = reverse(a);   reverse(a, 5);
}

int * reverse( int a[], int n ) {
    (or)
    int *pa, int n )
    // create a new array
    int r[n];  // Variable length array (VLAs).
    for( int i = 0; i < n; ++i )
        r[n-i-1] = a[i];

    return r;

}
```


reverse:
0x7000
r:
5
4
3
2
1
main
a:
1
2
3
4
5
p:
0x7000

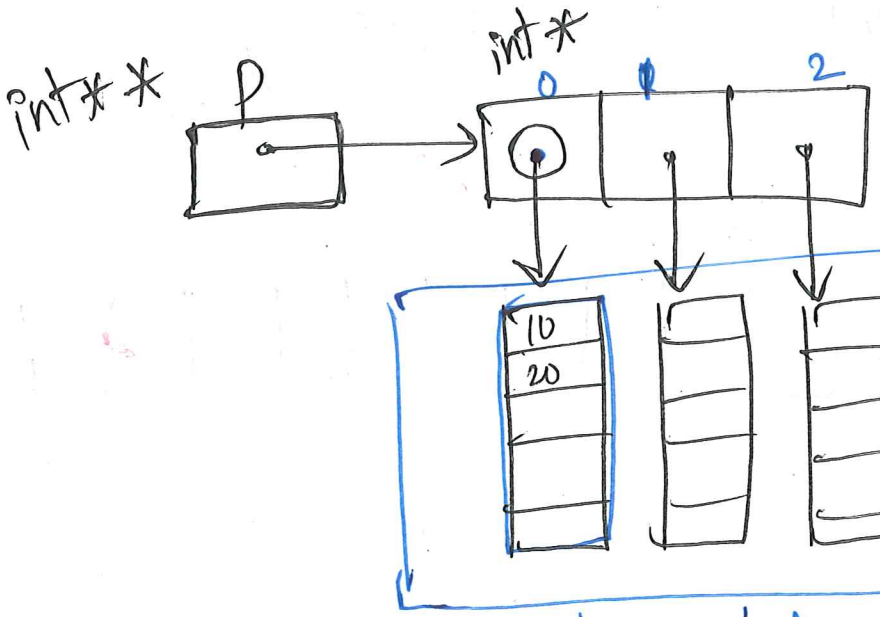
④

```
int * reverse(int a[], int n) {
    int *r = malloc(n * sizeof(int));
    // populate r.
    return r;
}
```

# of bytes



8192

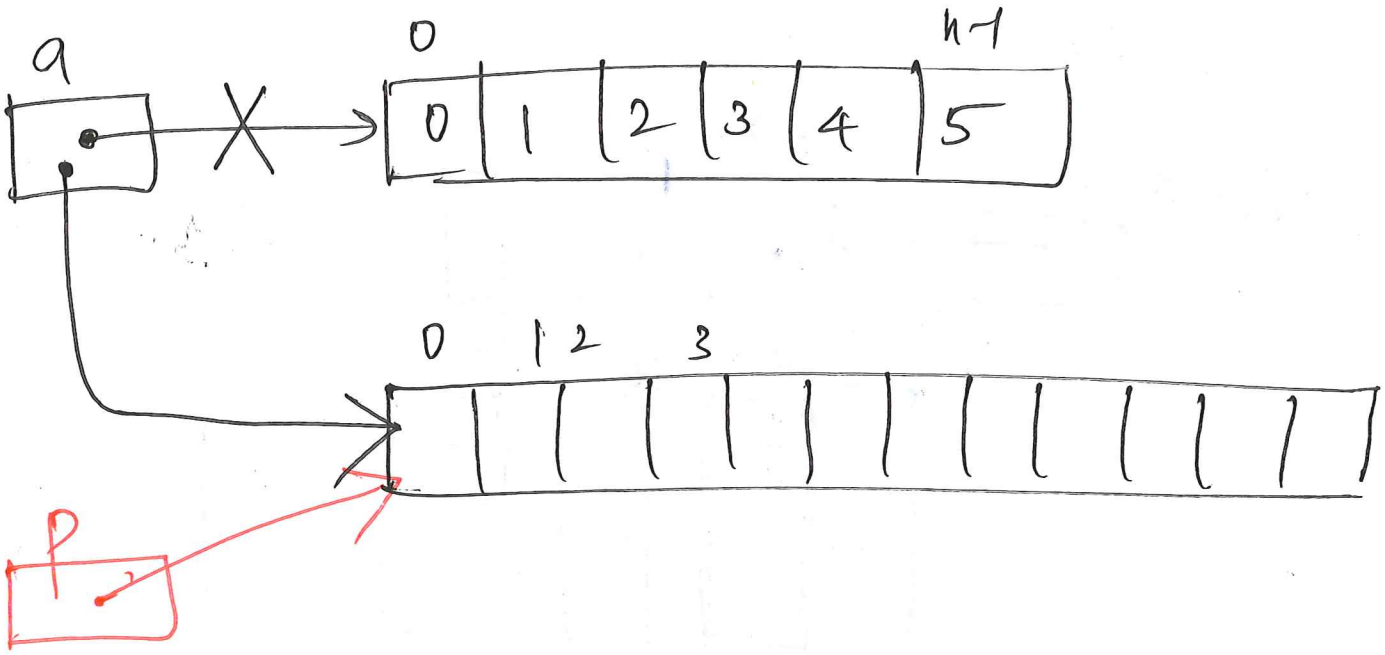


free(p)

before freeing p

loop:
free(p[i]);

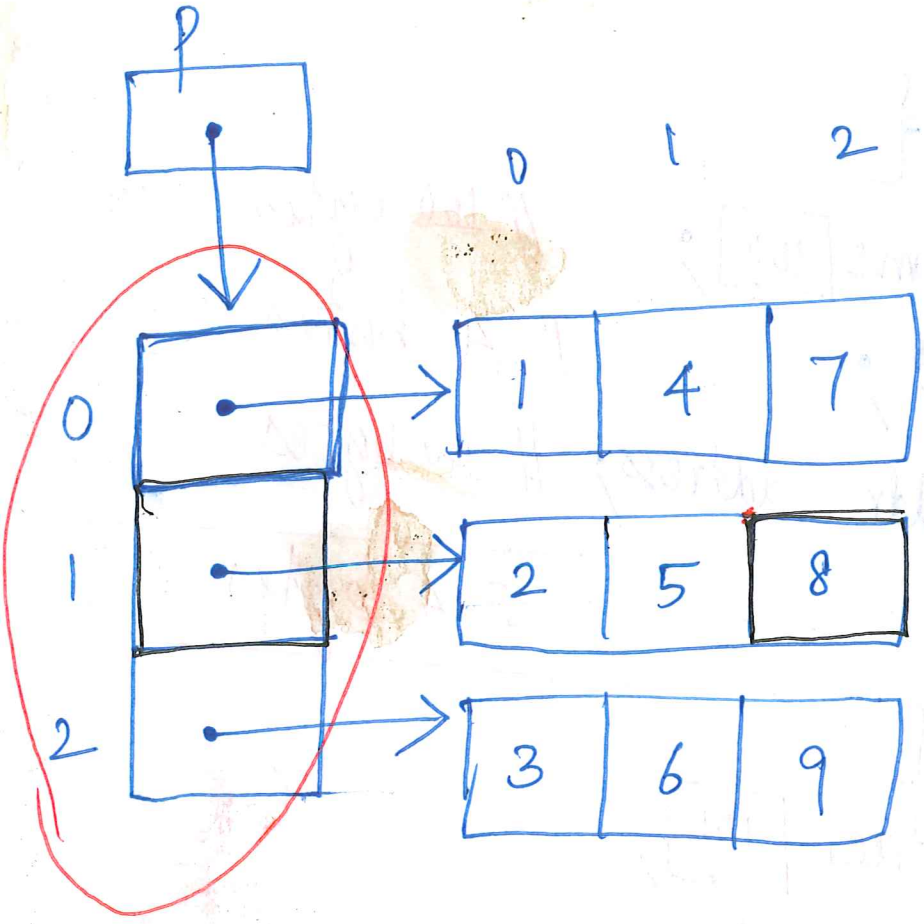
free(p);



$$a = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$t = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

(7)



p[i][j]

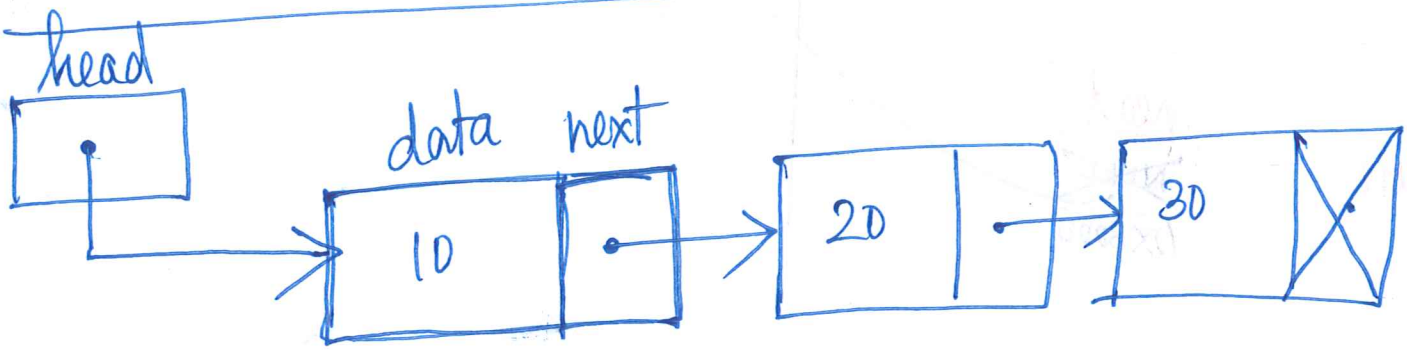
p[1][2] = 8

$$*(*(p+1)+2) = 8$$

Struct

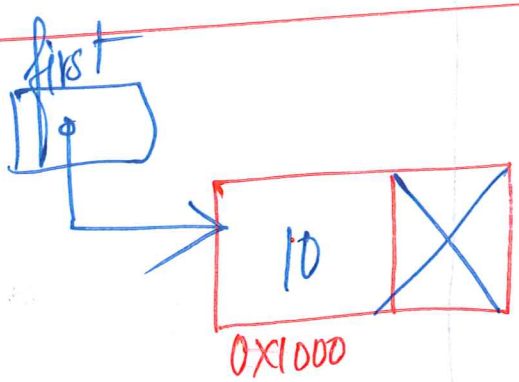
```
struct person {
    char name[100];      // 100 bytes
    int id;              // 4 bytes
    struct addr address; // 110 bytes
};                       // 214 bytes

struct addr {
    char street[100];
    char apt[10];
    int something;
};

struct person p1;
```

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

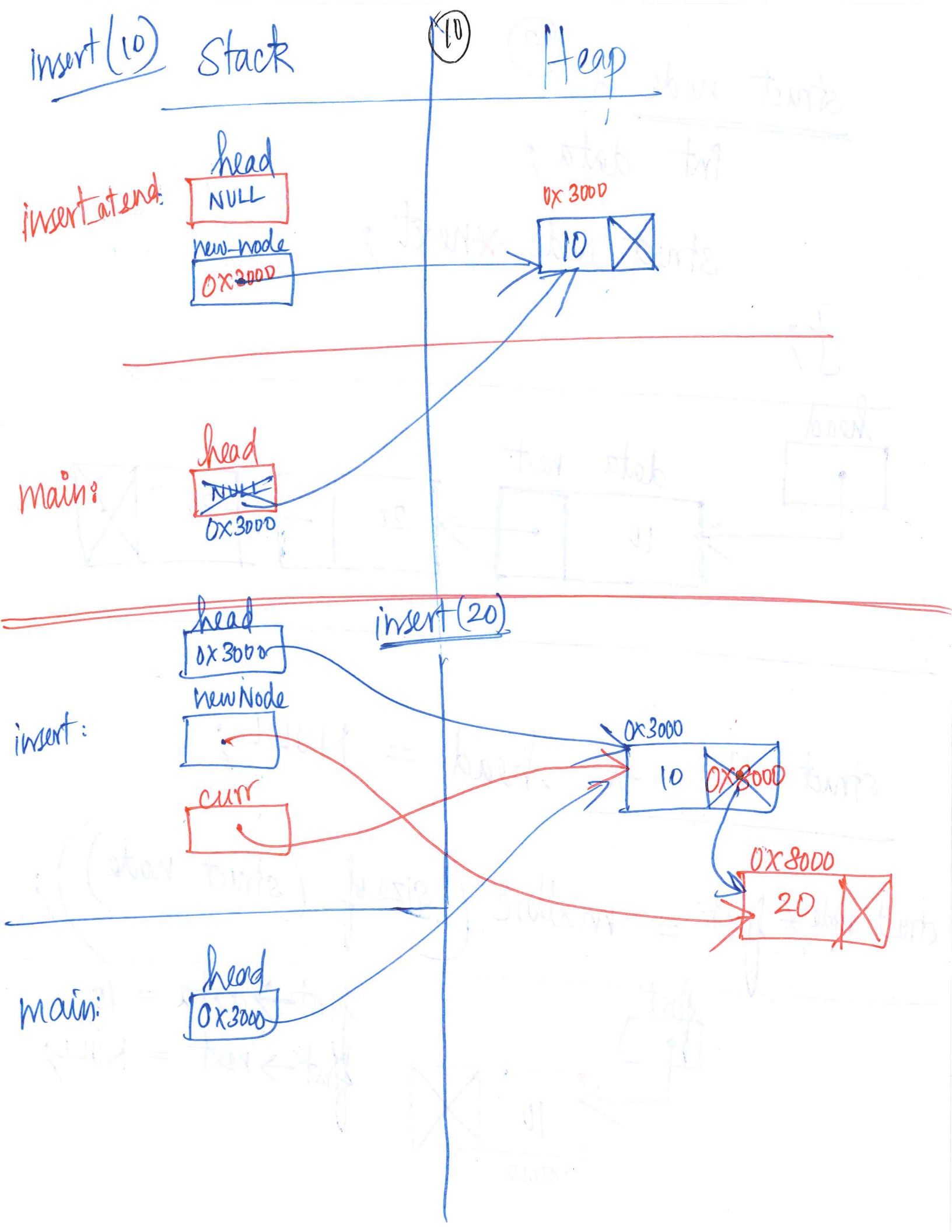


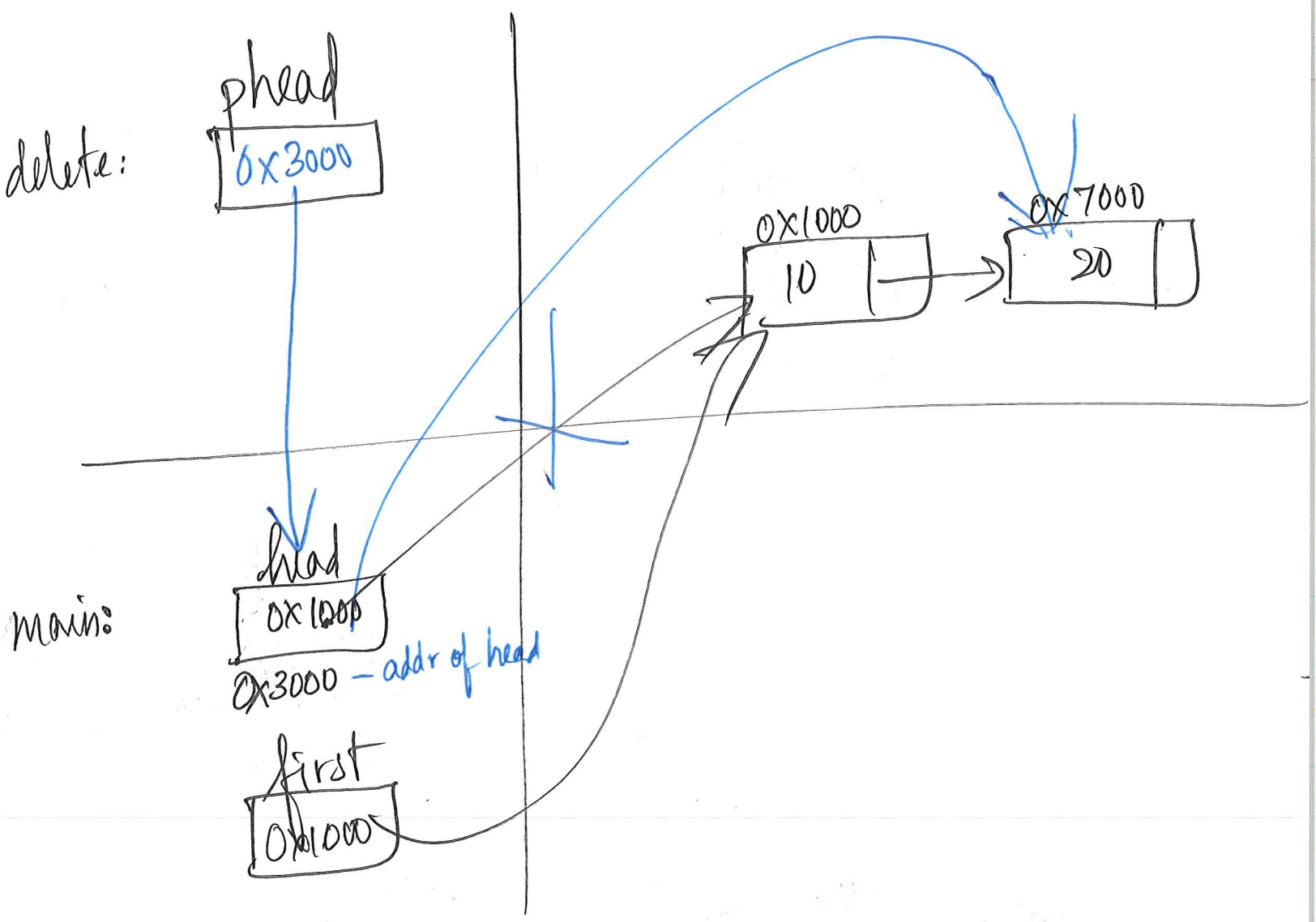

```
struct node * head = NULL;
```

```
struct node * first = malloc (sizeof (struct node));
```




```
first -> data = 10;
first -> next = NULL;
```


insert(10)
Stack
Heap
10
head
NULL
insert_at_end:
new_node
0x3000
0x3000
10
main:
head
NULL
0x3000
insert(20)
head
0x3000
newNode
insert:
curr
0x3000
10
0x8000
0x8000
20
main:
head
0x3000


delete:
phead
0x3000
0x1000
10
0x7000
20
main:
head
0x1000
0x3000 – addr of head
first
0x1000