Today

→ A bit more info on memory

→ Linked List
  - Create ✓
  - Length ✓
  - Print
  - Insert at End

→ Stack

→ Queue

→ Bonus stuff.

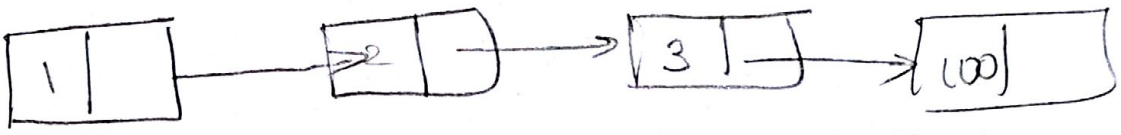
Print

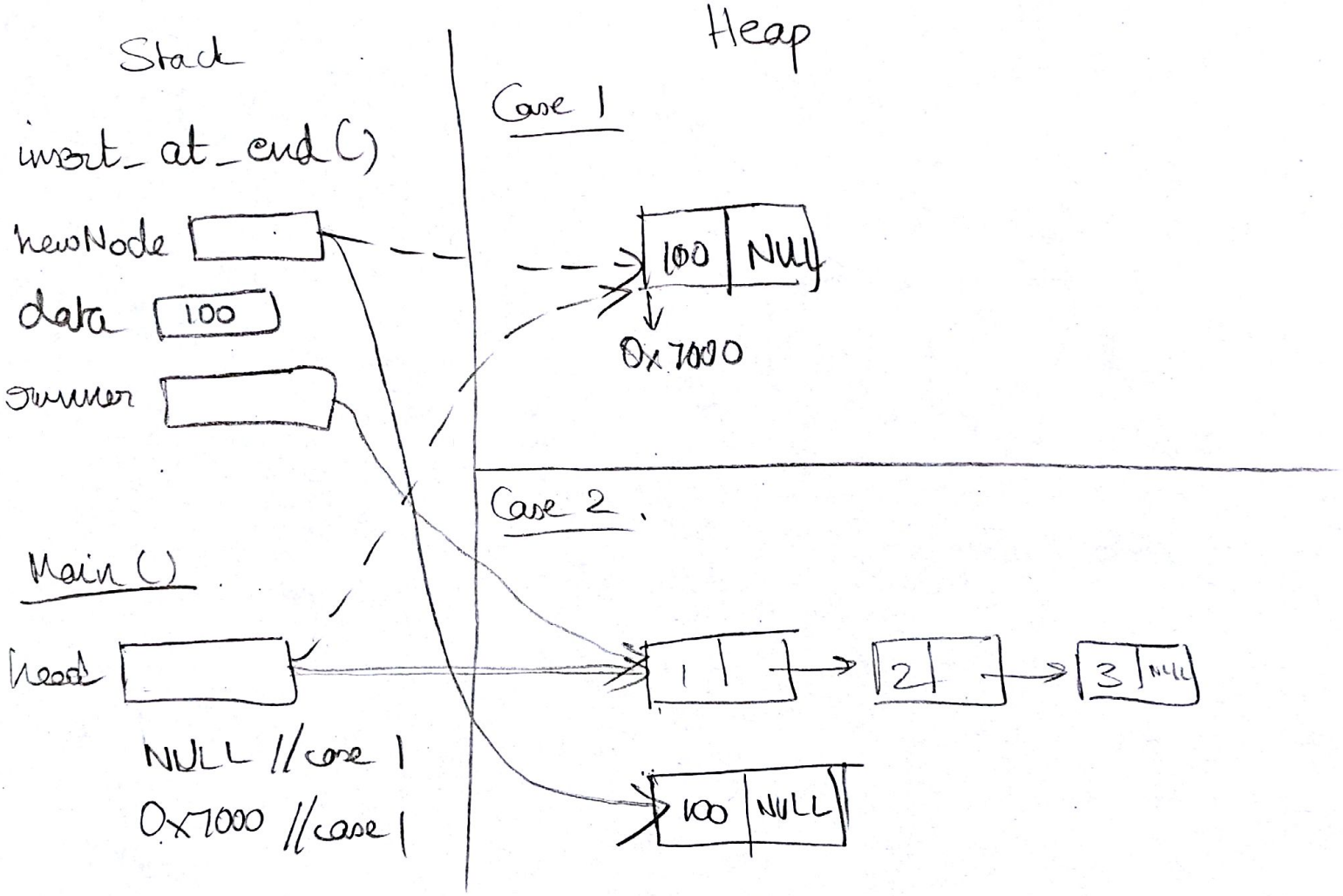
Show code

Memory Layout of a C program

```
#include
int number;
int main()
{
```

Insert at End






```
main ( )
{
    struct node * head = NULL;
    head = buildOneTwoThree ( );
    head = insert_at_end (head, 100);
}
```

Code :

```
struct node * insert_at_end (struct node * head,
                             int data)
{
    struct node * newNode = NULL;
    newNode = (struct node *) malloc (sizeof
                                      (struct node));

    newNode -> data = data;
    newNode -> next = NULL;

    struct node * runner = head;

    if (runner == NULL)                    // case 1
    {
        return newNode;
    }
    while (runner -> next != NULL)
    {
        runner = runner -> next;
    }
    // runner will point to the last node
                in the list
```

```
runner -> next = newNode;

return head;
}
```

Other List functions

Insert_at_front ( )

Insert_at_position (struct node *head,
int data,
int position)

Delete

Search

Stack

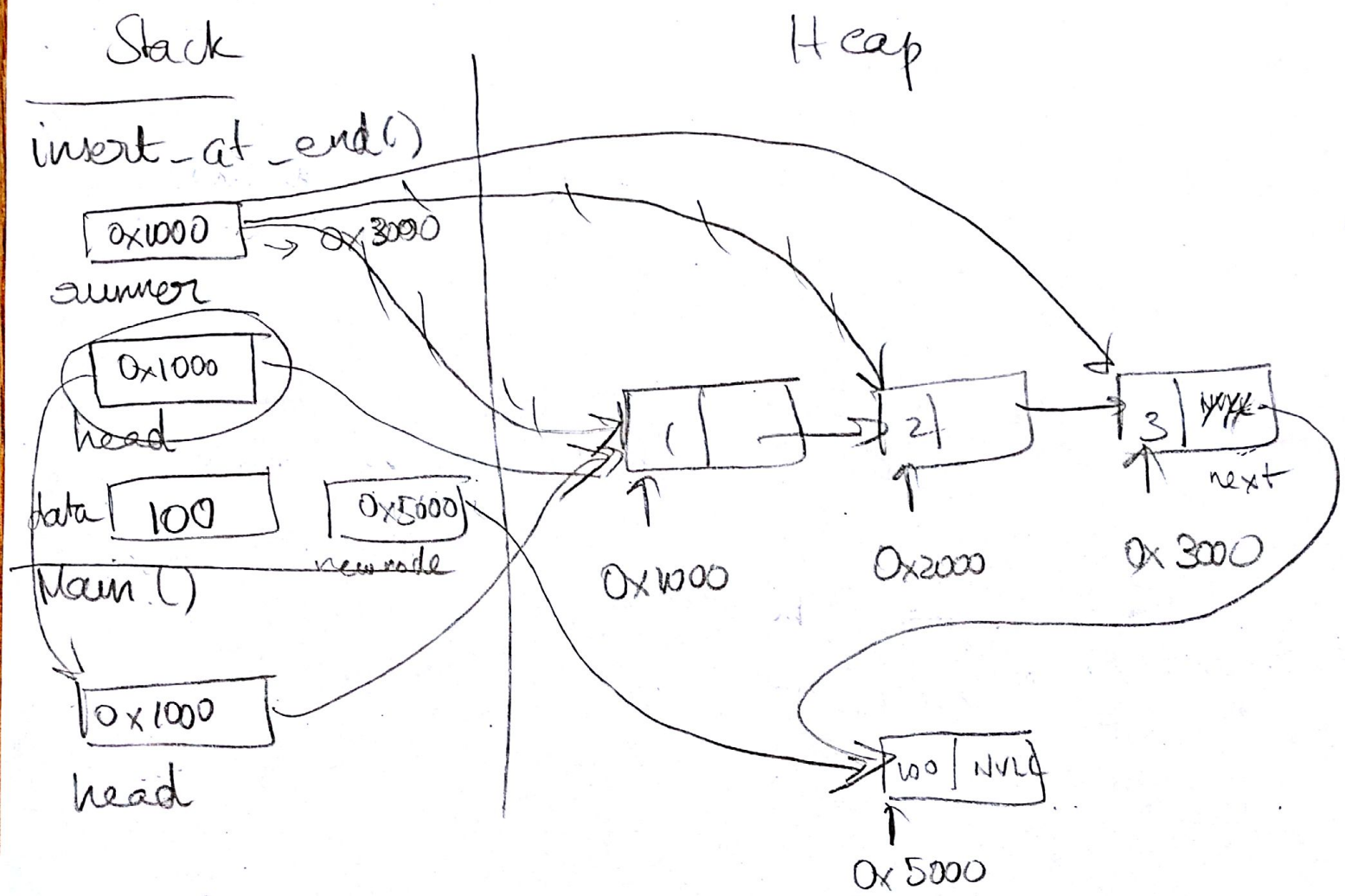
Last In First Out

Push Adds an element at the top.

or)

Insert at the head of the list.

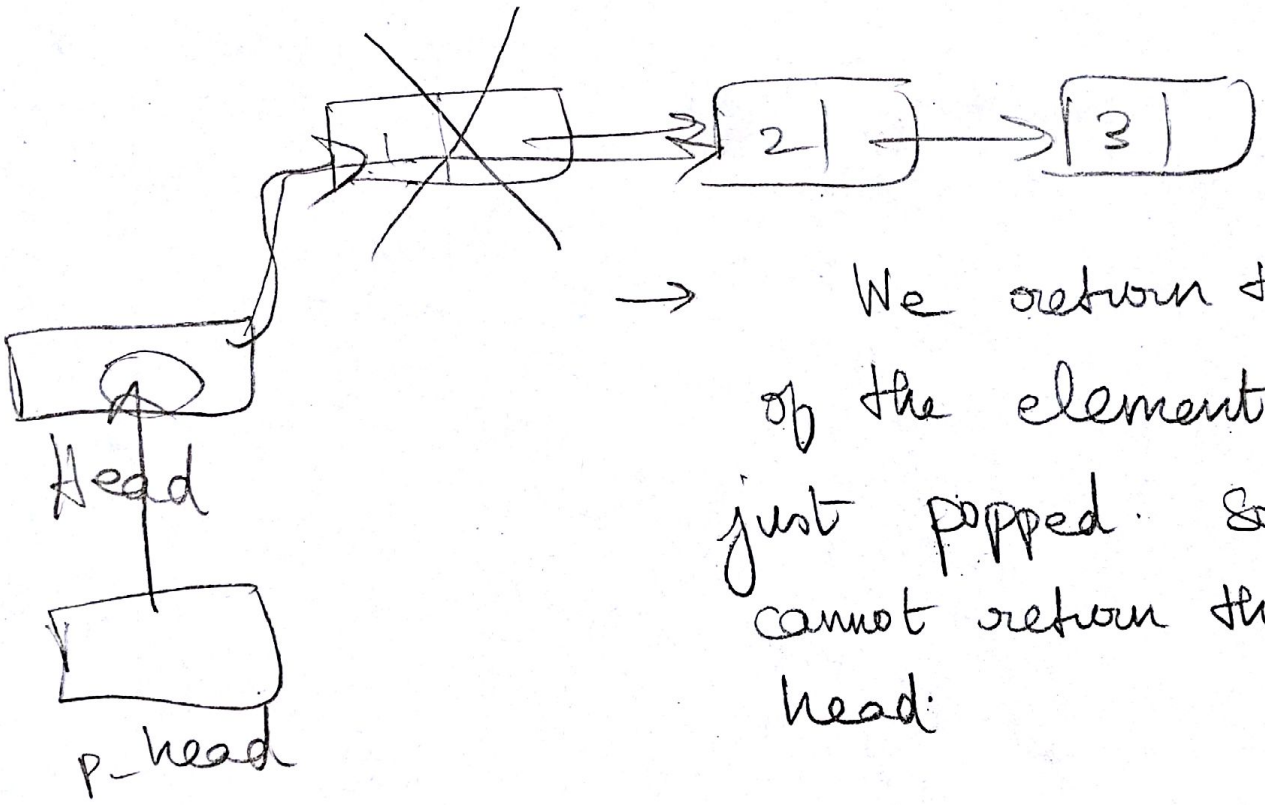
```
Struct node * push (struct node *head,
                    int data).
```



Pop

remove the head of the linked list and return its value.

int pop (struct node** head)



→ We return the value of the element we just popped. So we cannot return the new head.

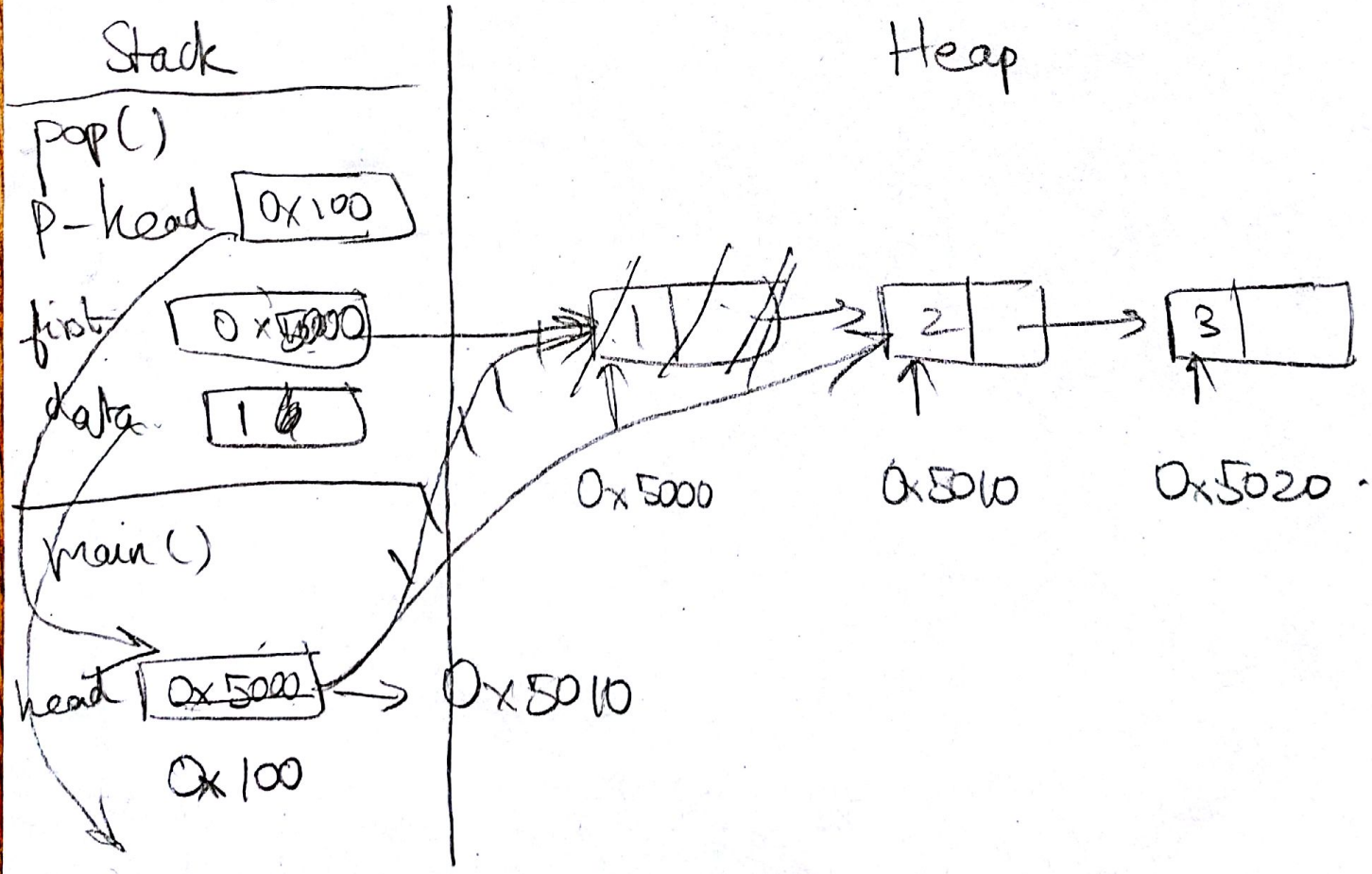
Solution?

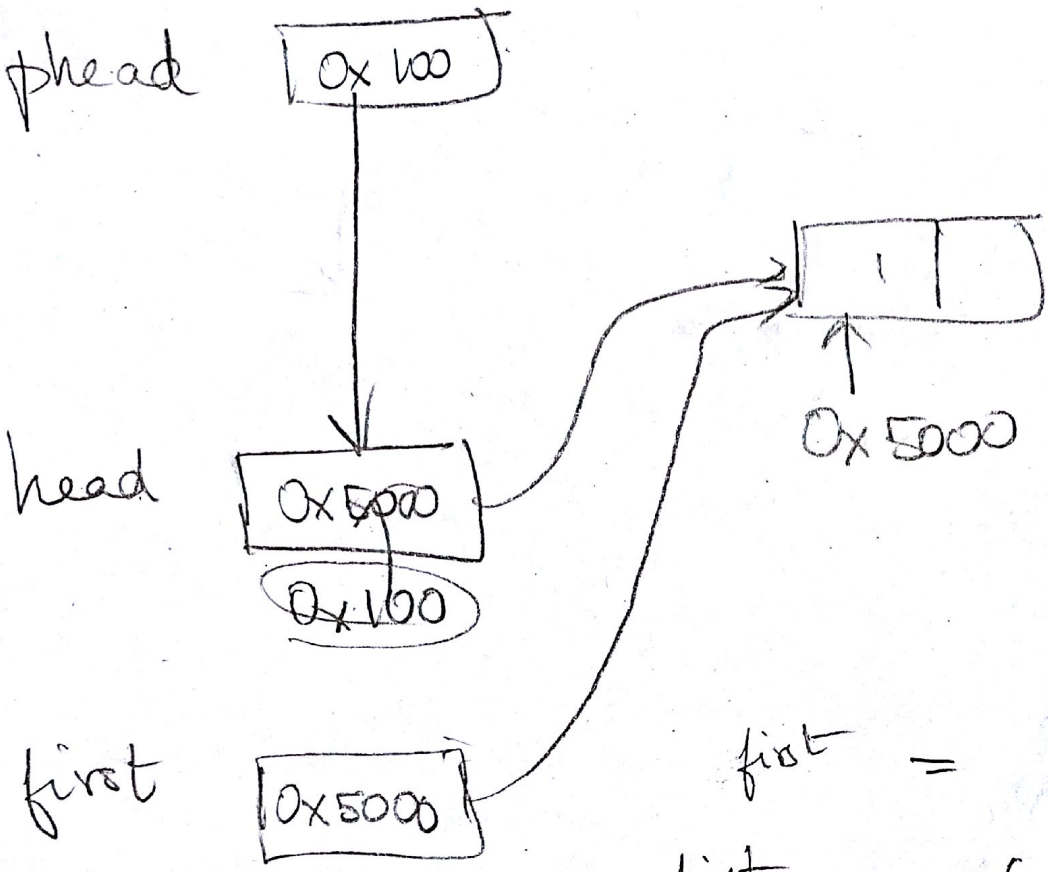
Pointer to the linked_list's head! (Ptr to a Ptr)

```
int pop (struct node ** p_head)
{
    struct node * first = * p_head;        (0x100)
                          ↘ value at (0x100)
    assert (first != NULL) //assert.h
      if (first == NULL)
            return -1;
    int data = first -> data;
    *phead = first -> next;
      free (first)
} return data;
```

```
main()
{
  int n = pop(&head)
```

```
int n = pop(&head)
```





first = * phead;

first = (value at 0x100)

first = 0x5000