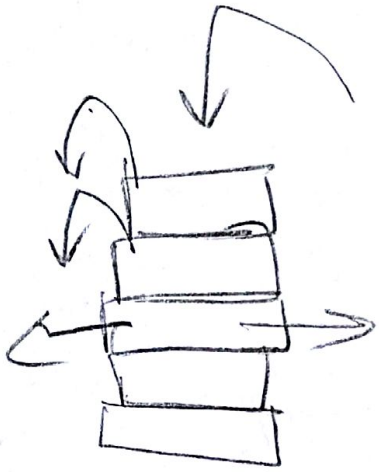
Announcements

- Quiz 1 out today! Due Sunday at 11:30 PM
- Tentative schedule for quizzes & projects.

Stack

A datastructure with specific rules for access (retreival) and insertion.



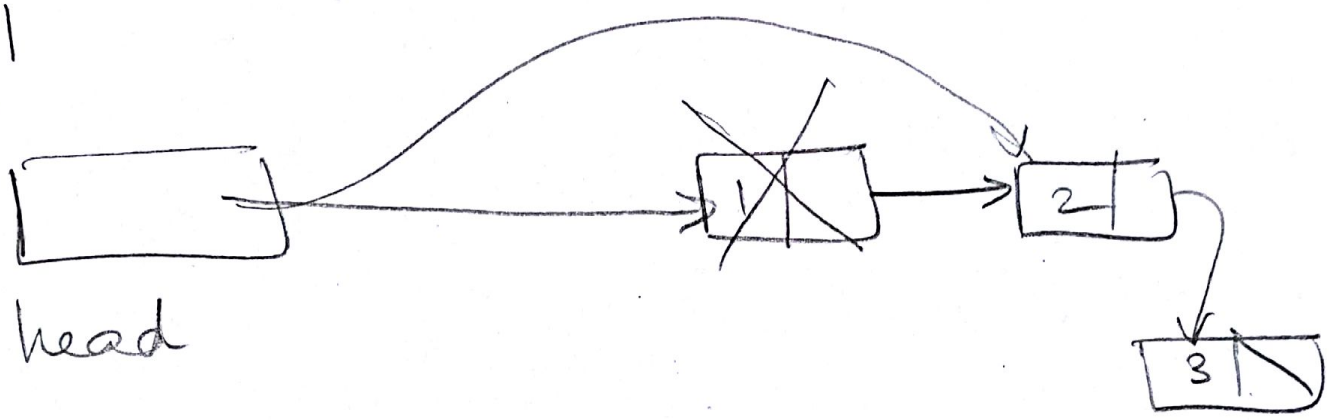
push → adds an element to the top of the stack (or) at the beginning of the list.

struct node * push (struct node * head, int data)

pop → removes the element at the beginning of the linked list

Stack

Heap



int pop (struct node **phead)

char *a = NULL;

a = (char *) malloc (100);

↑

Type cast

(void *) to (char *)

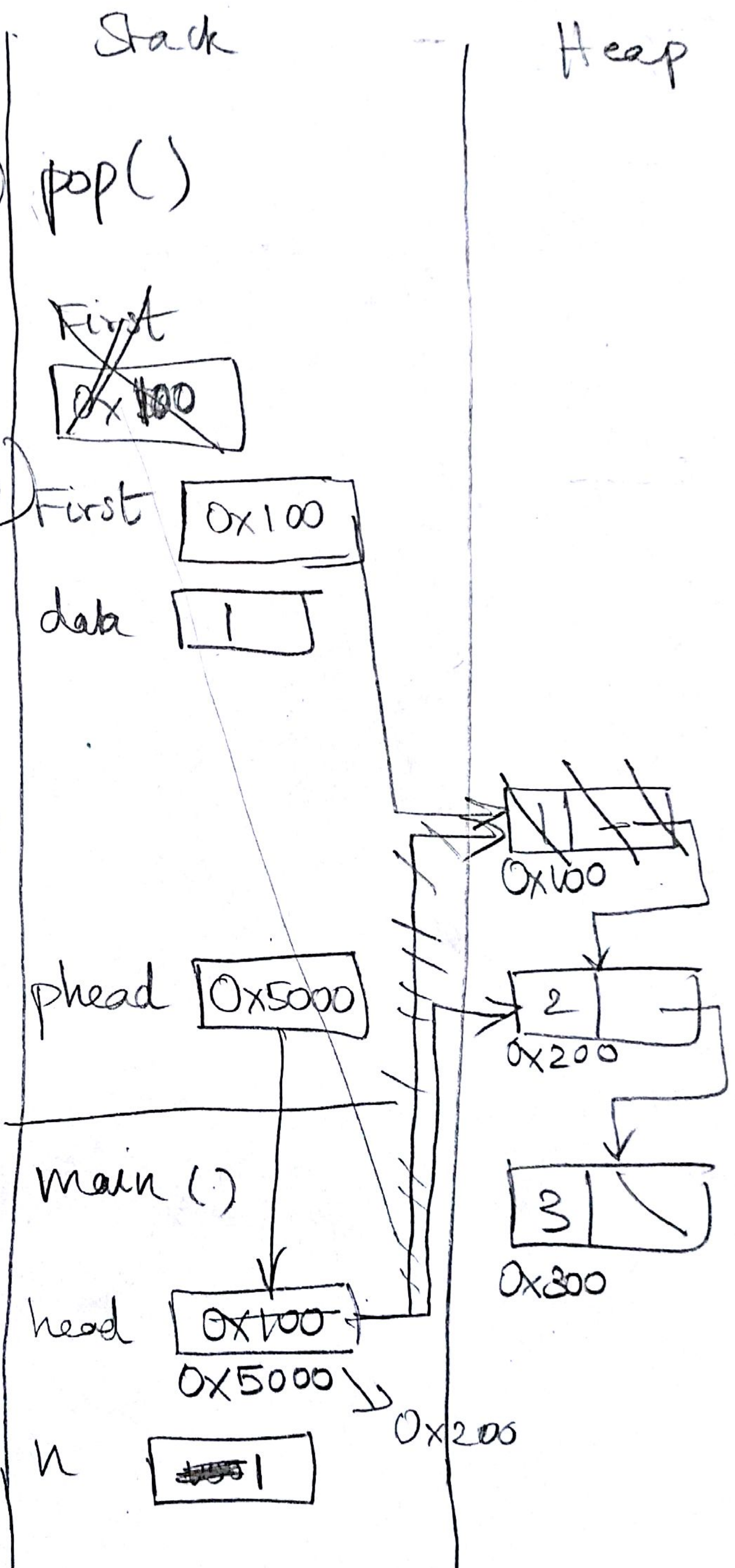
↓ returned by malloc

↓ Type of 'a'

Code

```
int pop (struct node
            **phead)
{
 struct node* first
            = *phead;
              (value at 0x5000)
 assert (first != NULL);
            //assert.h
 int data = first->data;
 *phead = first->next;
 free (first);
 return data;
}

int main ( )
{
  struct node * head
               = NULL;
   // add nodes to the stack
  int n;
//n = pop (&head);
  return 0;
}
```



Scanned by CamScanner

struct node * insert_at_end (struct node* head, int data)

↓ using a double pointer

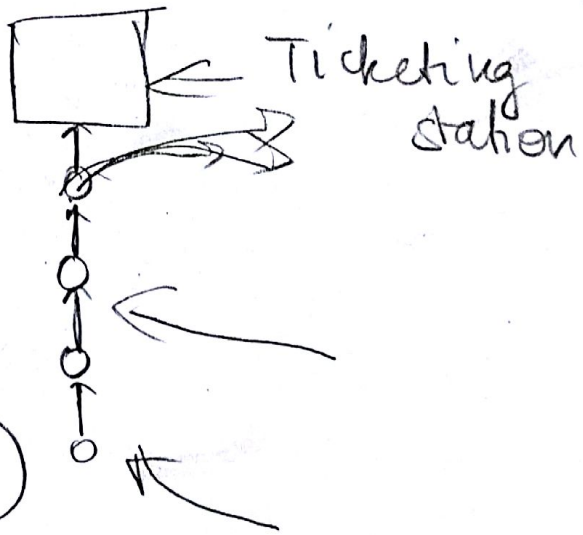
```
void insert_at_end(struct node **phead, int data)
{
    struct node *runner = *phead;
        // Make runner point to the first node.
    struct node *newNode = NULL;
newNode = (struct node *) malloc (sizeof(struct node));
newnode -> data = data;
newnode -> next = NULL;
//case 1 : When the list is empty
if (runner == NULL)
{ *phead = newNode;
  return;
}
```

// return newNode;

```
main()
{
    head = NULL;
insert_at_end
    (&head, 100);
}
```

```
case 2: // List is not empty -
while (runner -> next != NULL)
{
    runner = runner -> next;
}
// At this point, runner will point to the
                    last node.
runner -> next = newnode;
return;
}
```

Queue



enqueue → Adds an element to the end of the queue (insert_at_end)

dequeue → Removes ~~or~~ the element at the beginning of the queue

head 

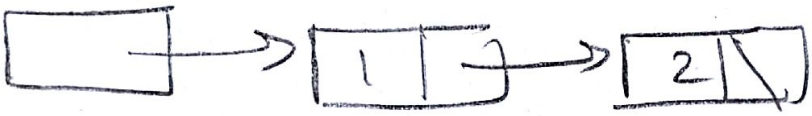
FIFO

Queue

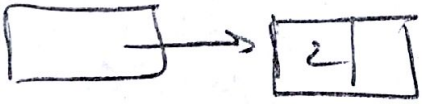
enqueue (&head, 1)

head 

enqueue (&head, 2)

head 

dequeue (&head) (pop)

head 

Preprocessor Directives → #define

→ #include

Macros: (See code)

A fragment of code that has been assigned a name.

#include <stdio.h>

#include "myheader.h"

main.c

```
#include "myheader.h"
int main()
{
    printf("%d", var);
    sayHello();
}

void sayHello()
{
    printf("Hello");
}
```

myheader.h

```
int var = 100
void sayHello();
struct node
{
    ___ ___
};
```