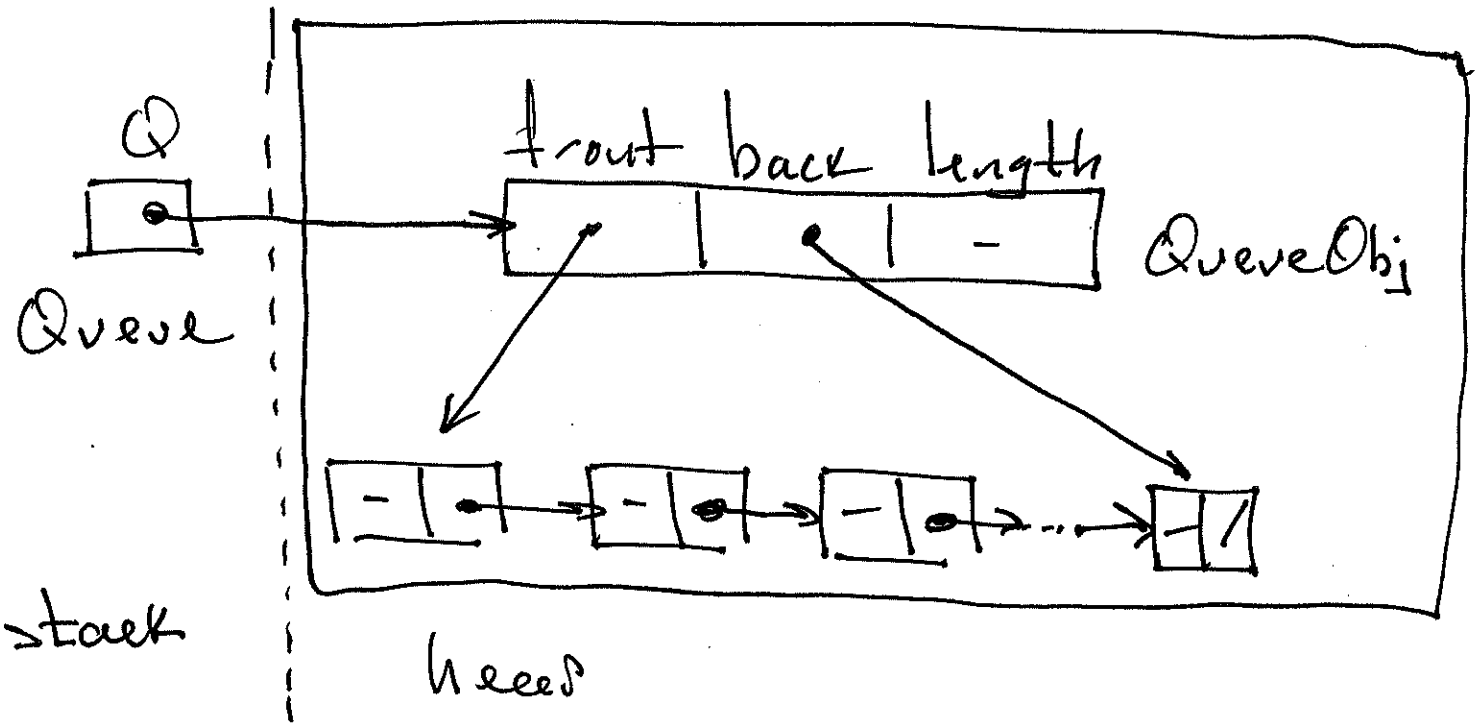
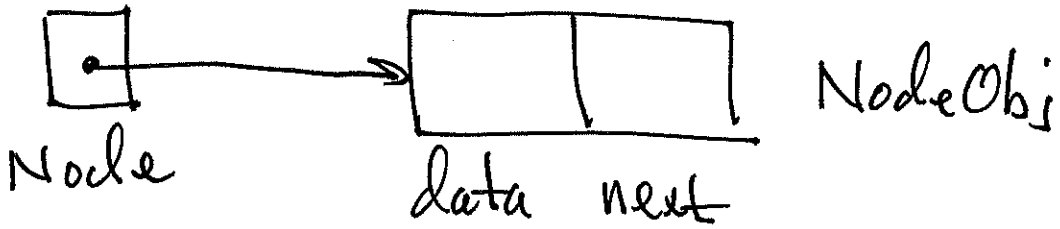
CSE 101 - 01 3-31-22

inside picture: (Queue example)





typedef A B ;

↑ this is an alias for A .

typedef struct NodeObj{ ← A

int data;

struct NodeObj* next;

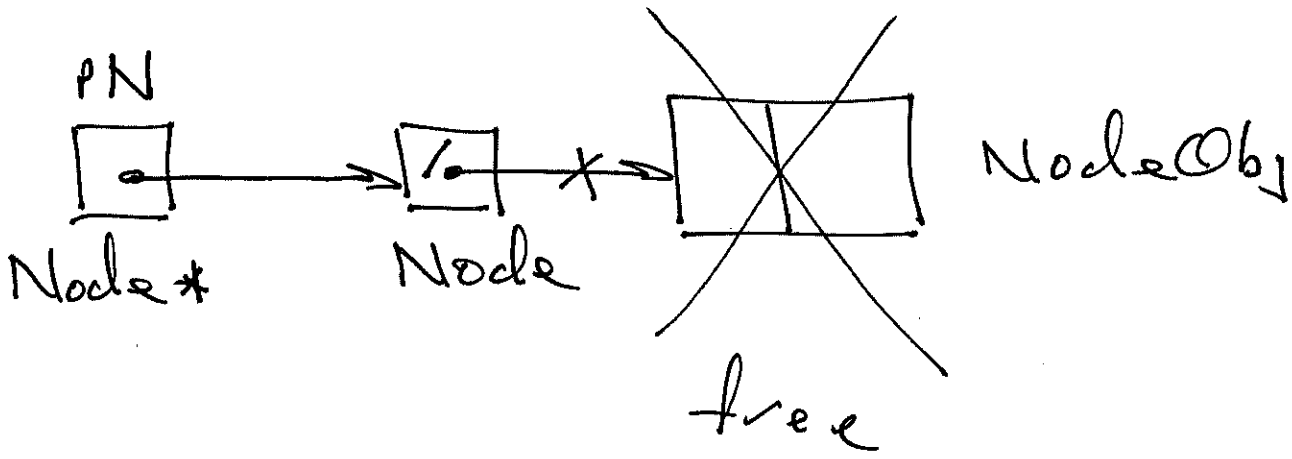
} NodeObj; ← B

typedef NodeObj* Node;

note: in C and C++:

(*N).data same as N->data

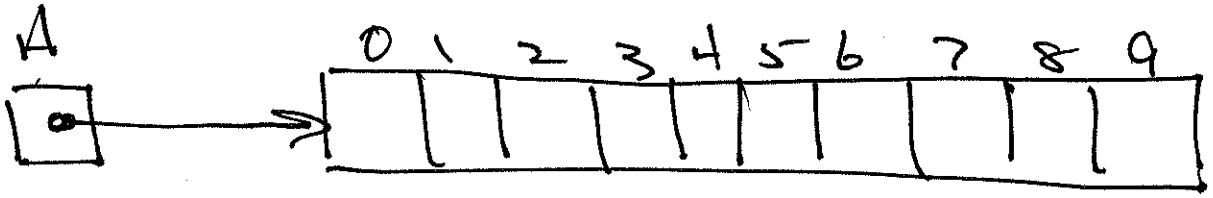
note: how to free a Node



Variable length Arrays (VLA) in C.

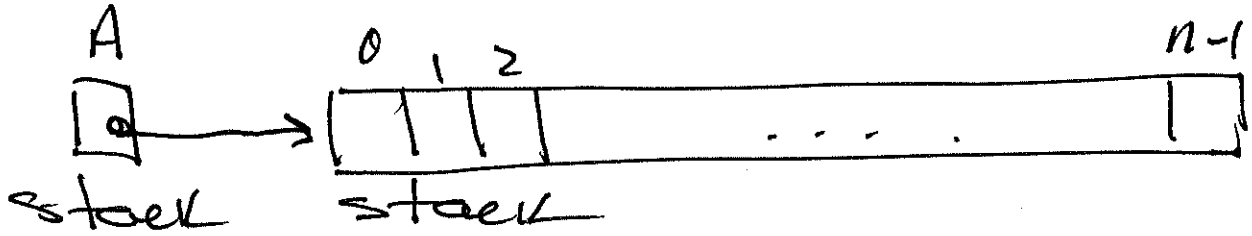
to create an array in C

```
int A[10];
```



Can also do

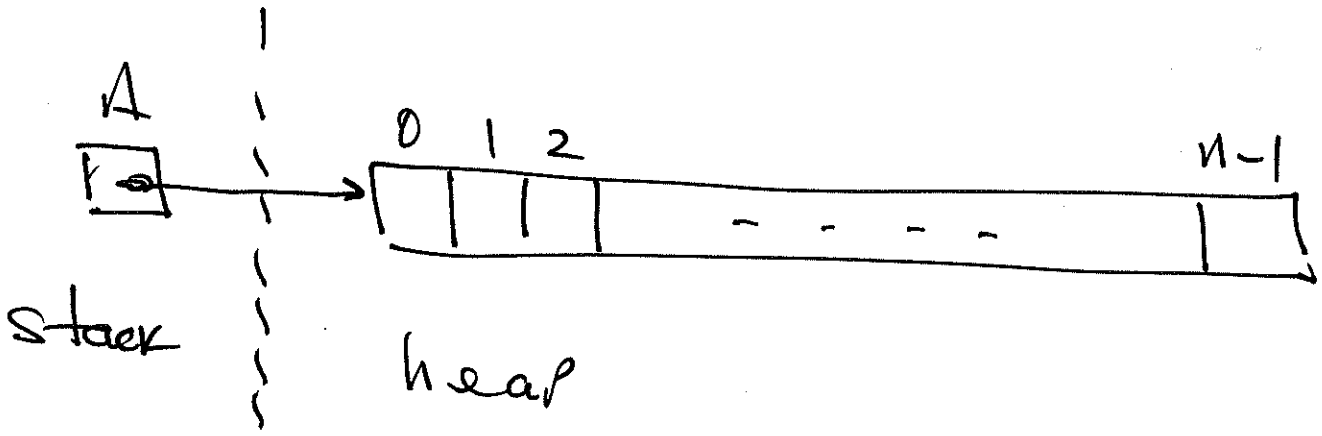
```
int n; // get n from input
int A[n];
```



Rule: Don't do this.

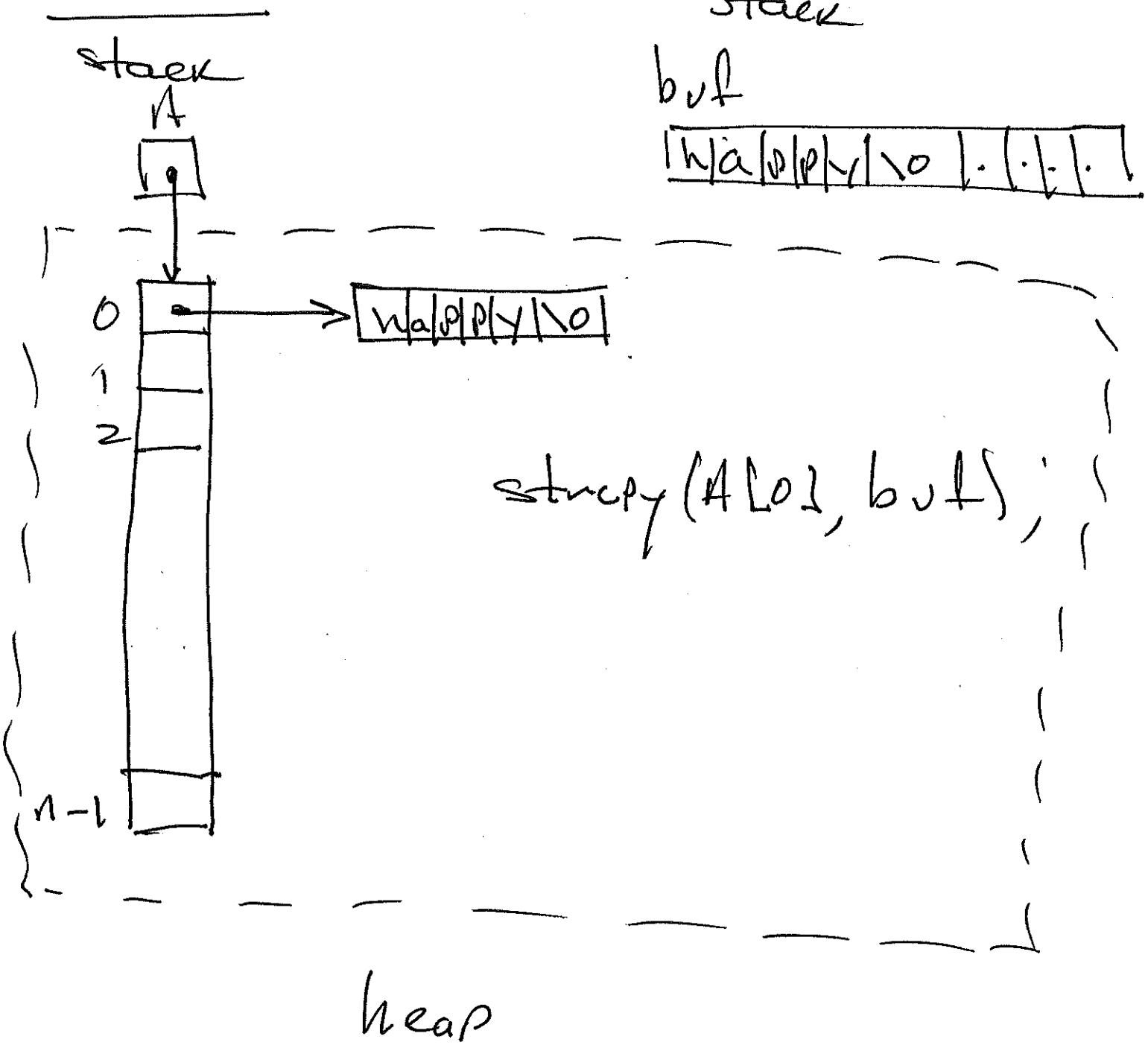
instead, use heap memory

```
int n; // get n from input
int* A;
A = calloc(n, sizeof(int));
```




```
// later
free(A);
```

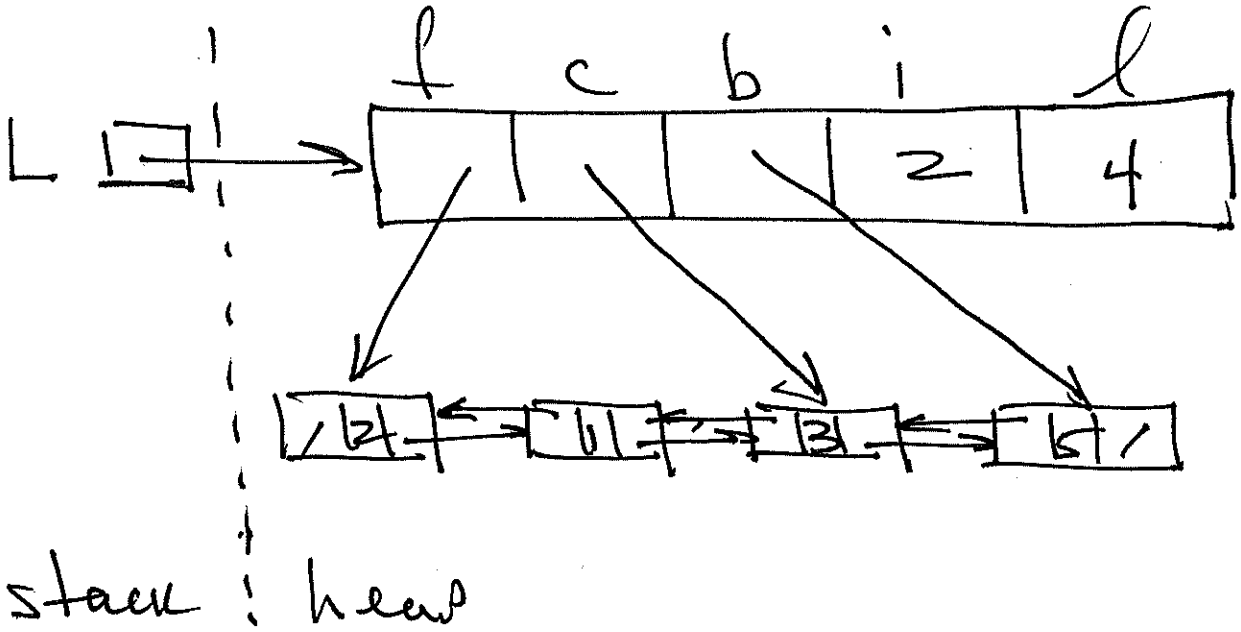
In Pal:



List ADT :

client view : (2, 1, 3, 5)
0 1 2 3

Inside view :



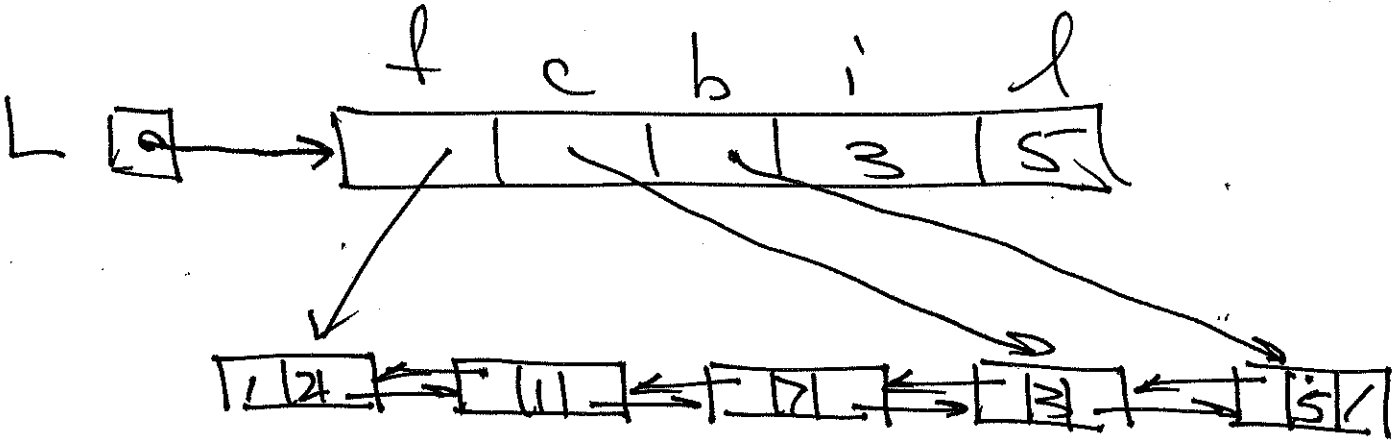
stack : heap

Do insertBefore(L, 7)

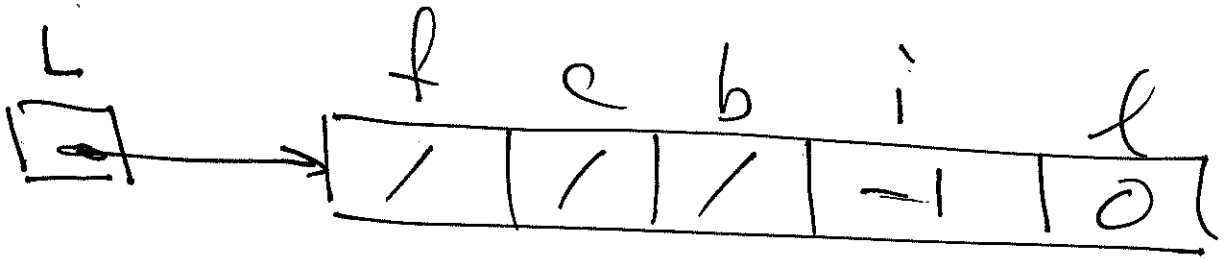
client: (2, 1, 7, 3, 5)

0 1 2 3 4

inside:



Empty state:



Pal example:

Suppose List ADT is built.

how to 'indirectly' sort an array of strings.

$A = (\underset{0}{\text{"c"}}, \underset{1}{\text{"a"}}, \underset{2}{\text{"b"}}, \underset{3}{\text{"d"}})$

want $L = (1, 2, 0, 3)$

start: $L = ()$

insert 0: $L = (0)$

insert 1: $L = (0)$ → insertBefore

$L = (1, 0)$

insert 2 : L = (1, 0)

L = (1, 0)

L = (1, 2, 0)

insert 3 : L = (1, 2, 0)

L = (1, 2, 0)

L = (1, 2, 0)

L = (1, 2, 0)

L = (1, 2, 0, 3)