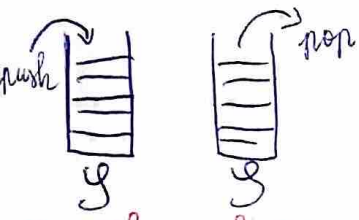
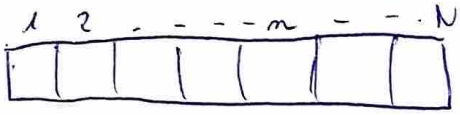
Stive



a => S S => a

void push(a) T pop(a) {returnează elementul șters}

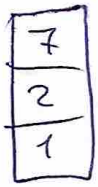


n = nr. elemente stivă

```
Push(a):
if n=N then
        write "OVERFLOW"
        stop
endif
n++;
x[n]=a;
```

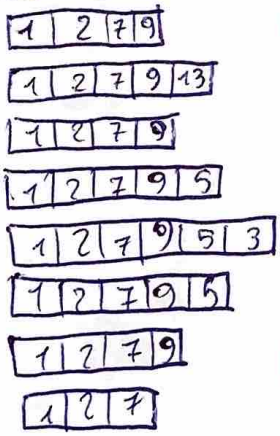
```
Pop(a):
if n=0 then
        write "underflow"
        stop
endif
write x[n] "este elemnt"
n--;
```

Se dă stiva următoare:

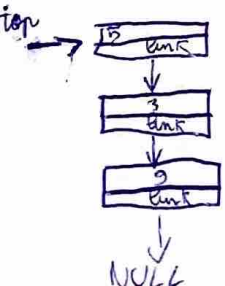


Să se scrie cum se modifică stiva după următoarele operații:

```
- push(9);
- push(13);
- pop();
- push(5);
- push(3);
- pop();
- pop();
- pop();
```



Alocare înlănțuită




```
struct Nod{
    T info;
    Nod * link;
};
Nod* top;
```

Operații de bază:

1) Accesare

```
if top≠NULL write top -> info
else "UNDERFLOW"
```

2) Inserarea (Push(a))

```
Nod* p;
p=New Nod;
if p=NULL -> "OVERFLOW" stop
endif
```

p->info = a;
p->link = top;
top = p;

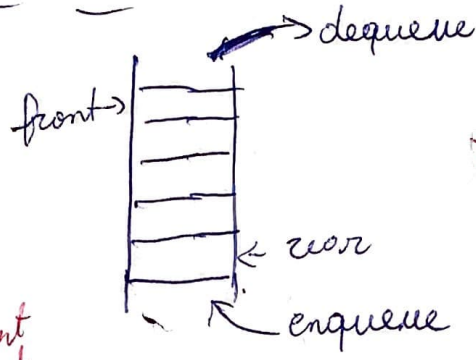
COZI

Fiecare element:

Nod* front; Nod* rear;

front => pointer prim element

rear => pointer ultim element

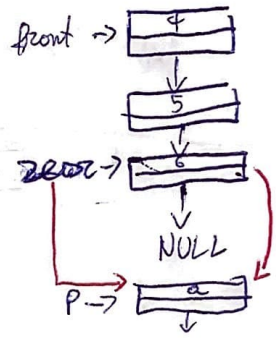


a => C enqueue(a)

C => a dequeue(a)

Operatii de bază

1) Inserarea unui element enqueue(a);




```
if p=NULL then {
    Write "OVERFLOW";
    Stop;
}endif
p->info = a;
p->link = NULL
if rear ≠ Null then
    rear->link = p;
endif else front = NULL;
rear = p;
```

2) Stergerea unui element dequeue(b)

```
if front==NULL && rear==NULL {
    cout << "UNDERFLOW";
    Exit;
}
temp = front;
front = front->link;
cout << "elementul sters este" << temp->info;
if front == NULL {
    rear = NULL;
};
delete temp;
```