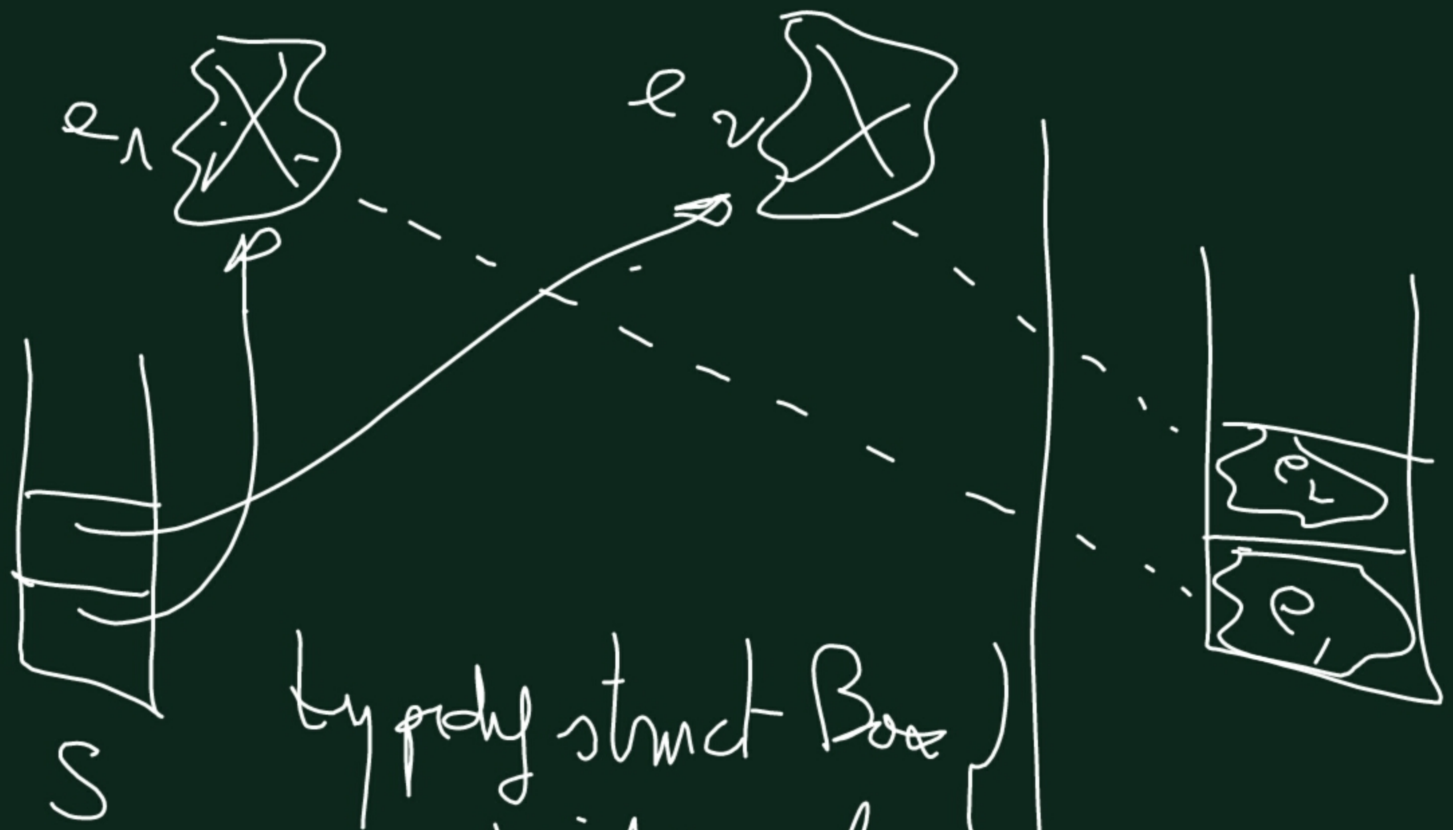


typedef struct Box {
    void *value;
    struct Box *next;
} Box;

Box *Stack;

/* Stack2.h — "owner" version */

```
typedef struct Box {
    struct Box * next;
    char value[1];
} Box;
```

```
typedef struct {
    Box * top;
    int eltSize;
    Box * available;
} Stack;
```

```
createStack(eltSize)
isEmpty
push
pop
top
disposeStack
```

```
Stack* s = createStack(sizeof(int));
```


```
Stack* createStack(int dtSize) {
    Stack* res = (Stack*) malloc(sizeof(Stack));
    if (!res) return 0;
    res->top = 0;
    res->dtSize = dtSize;
    res->available = 0;
    return res;
}
```


```
int isEmpty(Stack* s) {
    return !s->top;
}
```

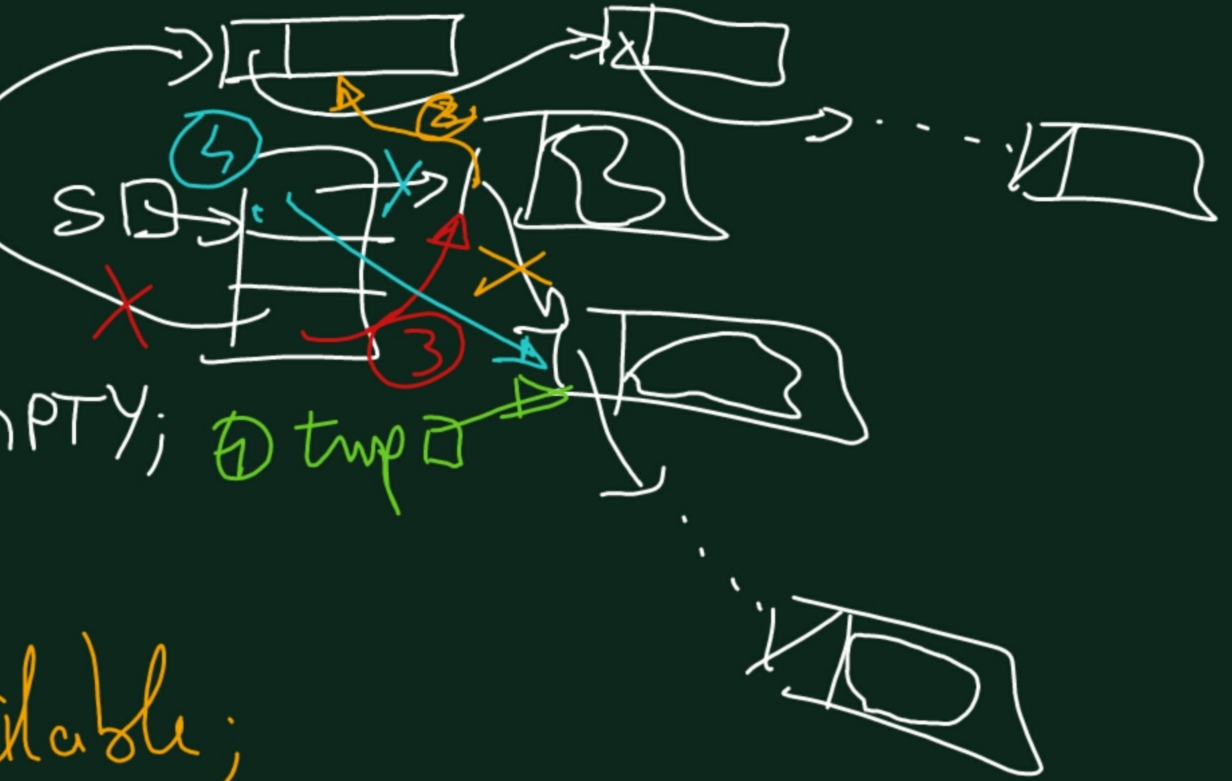
```
status push (Stack *s, void* e){
    status st;
    Box *temp = (Box*) malloc (sizeof(Box));
    if (!temp) return ERRALLOC;
    temp->next = s->top;
    s->top = tmp;
    memcpy(tmp->value, e, s->eltSize);
    return OK;
}
```

sizeof(Box) + s->eltSize - 1

```
if (s->available){ tmp = s->available;
                   s->available = s->available->next;
}
else ...
```

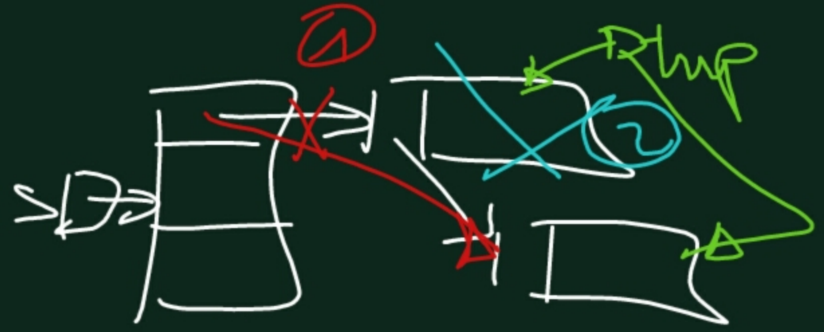
```
st=pop(s);

status Pop(Stack* s){
    Box *tmp;
    if(!s->top)return ERREMPTY;
    tmp = s->top->next;
    s->top->next = s->available;
    s->available = s->top;
    s->top = tmp;
    return OK;
}
```



```
status top (Stack * s, void* res){
    if(!s->top) return ERRENTITY;
    memcpy(res, s->top->value,
                s->elemsize);
    return OK;
}
```

```
void destroyStack(Stack *s){
    Box* tmp = s->top;
    while(tmp){
        s->top = tmp->next;
        free(tmp);
        tmp = s->top;
    }
}
```




```
tmp = s->available;
while(tmp){
    s->available = tmp->next;
    free(tmp);
    tmp = s->available;
}
free(s);
```

- create 2 ints (1 and 2) and 1 stack
- push ints onto stack
- while stack not empty
  - print top
  - pop stack
- destroy stack

expected output:

2

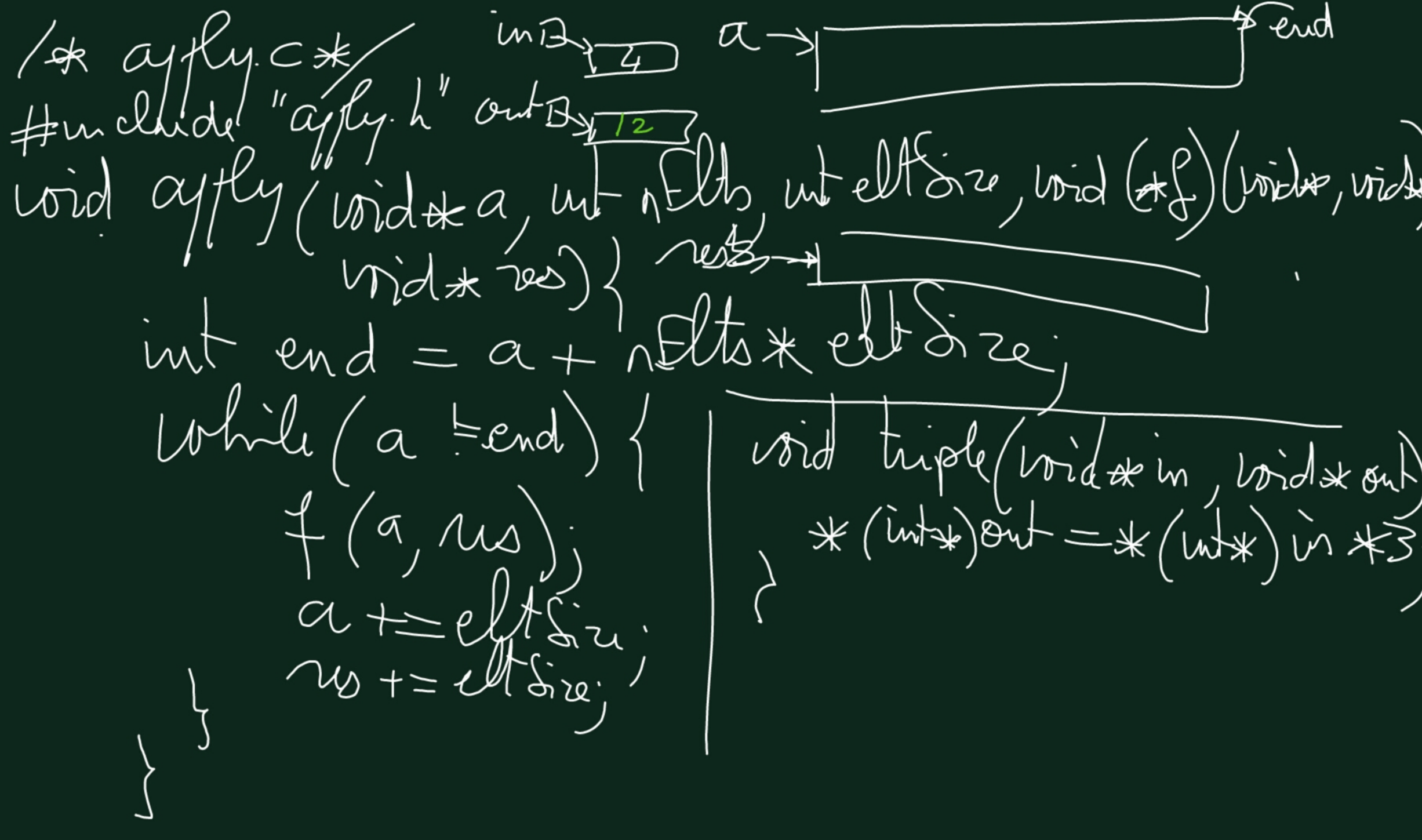
1

```
int e1 = 1, e2 = 2;
Stack* s = createStack(sizeof(int));
if (!s) return 1;
if ( st = push(s, &e1) ) {
    printf("%s\n", message(st));
    return 2;
}
if ( ____________ &e2 ) {
}
while ( !isEmpty(s) ) {
    int e;
    if ( st = top(s, &e) ) {
        printf( ____________ );
        return 3;
    }
    if ( st = pop(s) ) {
        printf( ____________ );
        return 4;
    }
}
destroyStack(s);
return 0;
```

https://tiny.one/examC1601

4:10

```
/* apply.h */
void apply(void *a, int nElts, int eltSize,
           void (*f)(void*, void*), void* res);
```


/* ayfly.c */
#include "ayfly.h"
void ayfly(void *a, int nElts, int eltSize, void (*f)(void*, void*),
void *res) {
int end = a + nElts * eltSize;
while (a != end) {
f(a, res);
a += eltSize;
res += eltSize;
}
}
inB
4
outB
12
a
end
res
void triple(void *in, void *out) {
(int)out = *(int*)in * 3;
}