

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "sqstack.h"
```

```
sqstack * stack_create(int len) {
    sqstack * s;
```

```
    if ((s=(sqstack *)malloc(sizeof(sqstack))) == NULL) {
        printf("malloc sqstack failed\n");
        return NULL;
    }
```

```
    if ((s->data = (data_t *)malloc(len * sizeof(data_t))) == NULL) {
        printf("malloc data failed\n");
        free(s);
        return NULL;
    }
```

```
    memset(s->data, 0, len*sizeof(data_t));
    s->maxlen = len;
    s->top = -1;
```

```
    return s;
}
```

```
int stack_push(sqstack * s, data_t value) {
```

```
    if (s == NULL) {
        printf("s is NULL\n");
        return -1;
    }
```

```
    if (s->top == s->maxlen-1) {
        printf("stack is full\n");
        return -1;
    }
```

```
    s->top++;
    s->data[s->top] = value;
```

```
    return 0;
}
```

```
/*
 * @ret 1-empty
 * */
```

```
int stack_empty(sqstack *s) {
    if (s == NULL) {
        printf("s is NULL\n");
        return -1;
    }
    return (s->top == -1 ? 1 : 0);
}
```

```
/*
 * @ret 1-full
 * */
```

```
int stack_full(sqstack *s) {
    if (s == NULL) {
```

sizeof 接判
malloc
sizeof 接判 (s)
free(s)

memset
data
memset
maxlen =
top =

top + 1
top = value

return
top - 1
0

XF : 且记: null

0
return
free

```

        printf("s is NULL\n");
        return -1;
    }
    return (s->top == s->maxlen-1 ? 1 : 0);
}

```

return.
 → top max-1 | 1
 0

```

data_t stack_pop(sqstack *s) {
    s->top--;
    return (s->data[s->top+1]);
}

```

→ top--
 return top+1

```

data_t stack_top(sqstack *s) {
    return (s->data[s->top]);
}

```

→ return data-top

```

int stack_clear(sqstack *s) {
    if (s == NULL) {
        printf("s is NULL\n");
        return -1;
    }

    s->top = -1;
    return 0;
}

```

→ top = -1

```

int stack_free(sqstack *s) {
    if (s == NULL) {
        printf("s is NULL\n");
        return -1;
    }

    if (s->data != NULL)
        free(s->data);
    free(s);

    return 0;
}

```

→ free → s->data (L_前 L_后) (非NULL)
 free → s (L_前 L_后) (非NULL)