A stack is a sequence of elements of the same type

\[ S_t = \langle a, r, p, l, q \rangle \]

**Operations**

- `top(S_t) = 'q'`
- `push(S_t, x) = \langle a, r, p, l, q, x \rangle`
- `pop(S_t) = \langle a, r, p, l \rangle`

**Example application:**

```
read string character by character 'c'
if 'c' is an opening symbol
    push 'c' into stack
if 'c' is a closing symbol
    compare 'c' to top of stack.
    if 'c' does not match top of stack
        ERROR!!
    else if 'c' matches top of stack
        pop the stack
if stack is empty SUCCESS!
```

Application:

```
main()  foo()  bar()
```

run stack
`call_stack`

```
"Stack overflow"
```
The ArrayStack Data Structure.

```java
// @param x the value to push onto the stack
void push(int x) {
    if (isFull())
        throw new StackOverflowError();
    m_data[m_size++] = x;
}

int pop() {
    if (isEmpty())
        throw new UnderFlowError();
    return m_data[--m_size];
}
```

The LinkedStack Data Structure.

```java
// @param x the value to push onto the stack
void push(int x) {
    Node tmp = new Node(x, m_head);
    m_head = tmp;
    m_size++;
}

int pop() {
    if (isEmpty())
        throw new UnderFlowError();
    int top = m_head->m_data;
    Node tmp = m_head;
    m_head = tmp->m_next;
    return top;
}
```
delete m-head;
m-head = tmp;
M-size--;