Announcements

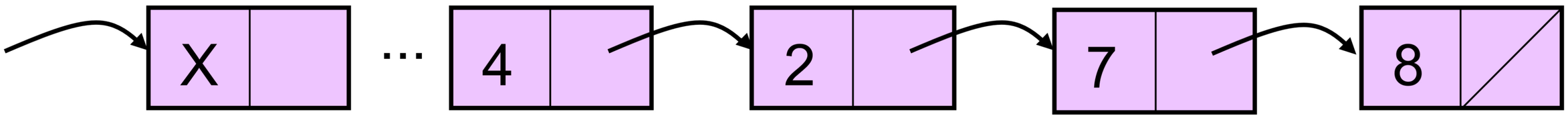
- MP3 due on 02/26 @11:59pm

- Exam 2: 03/02-03/04. The same process as last time, more interesting content, worth 15% of the final grade.

- If you'd like to revisit some of the concepts of the MPs, labs, or any lecture material, you are welcome to get help at **Pencil and Paper (PnP) office hours. These might also include some practice exam questions.**
https://chara.cs.illinois.edu/cs225/calendar/

Emily Chao: Mon 4-6pm, Siebel 0224/0226
Justin: Tue 4-6pm
Victor: Wed 3-5pm
Tristan: Fri 4-6pm
Rohan: Sat 12-2pm

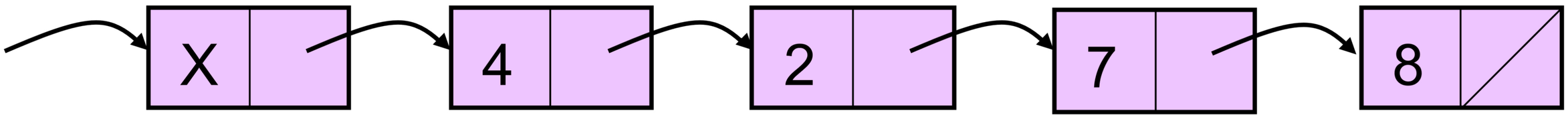
Remove node in fixed position (given a pointer to node you wish to remove):



Solution #1:

```
void List<LIT>::removeCurrent(listNode * curr) {
}
```

Remove node in fixed position (given a pointer to node you wish to remove):



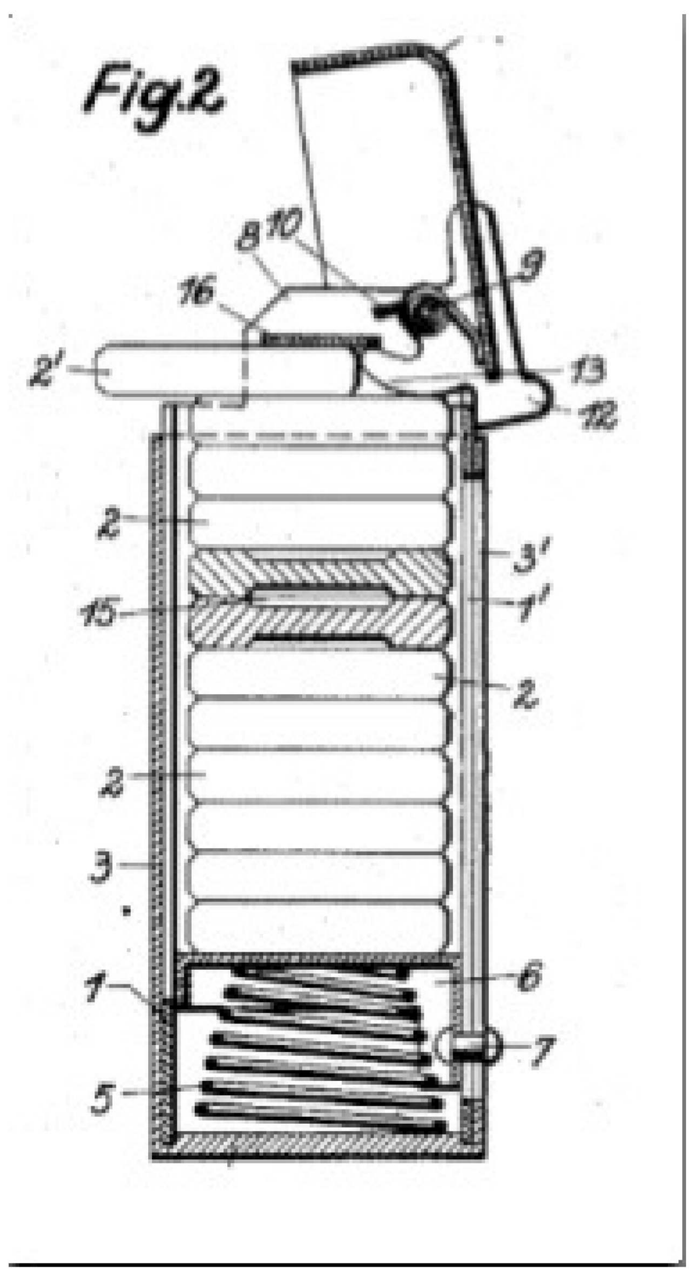
Constant time hack:

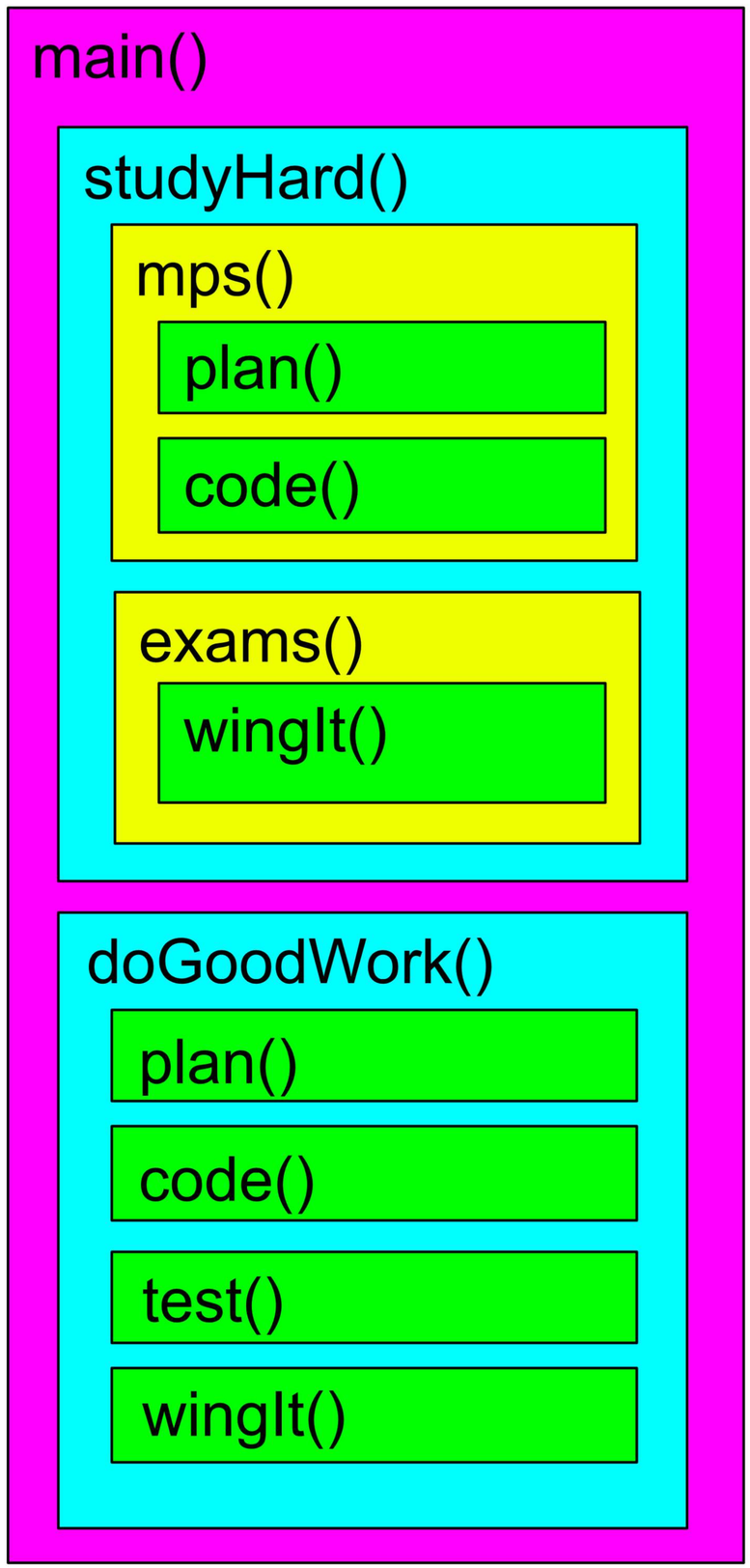
```
void List<LIT>::removeCurrent(listNode * curr) {

}
```

Summary – running times for List functions:

	SLL	Array
Insert/Remove at front:	O(1)	O(1)
Insert at given location:	O(1)	O(1)
Remove at given location:	O(1) hack	O(n) shift
Insert at arbitrary location:	O(1)	O(n) shift
Remove at arbitrary location:	O(n) find	O(n) shift

Stacks:





({ } ( ) [(( )) { (( )) ( ) }] ( ))

4 5 + 7 2 - * 3 - 6 /

Stack ADT:

```
template<class SIT>
class Stack {
public:
     Stack();
     ~Stack(); // also copy
     constructor, assignment op
     bool empty() const;
     void push(const SIT & e);
     SIT pop();
private:
```

```
};
```

push(3)

push(8)

push(4)

pop()

pop()

push(6)

pop()

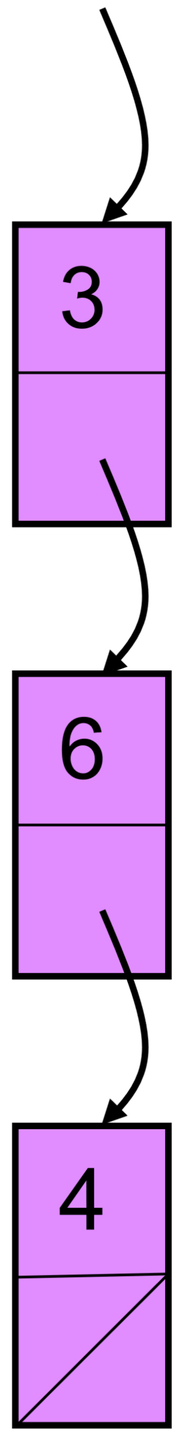
push(2)

pop()

pop()

Stack linked memory implementation:

```
template<class SIT>
class Stack {
public:
    Stack();
    ~Stack(); // etc.
    bool empty() const;
    void push(const SIT & e);
    SIT pop();
private:
    struct stackNode {
        SIT data;
        stackNode * next;
    };
    stackNode * top;
    int size;
};
```




```
template<class SIT>
SIT Stack<SIT>::pop(){




}
```

```
template<class SIT>
void Stack<SIT>::push(const SIT & d){
stackNode * newNode = new stackNode(d);
newNode->next = top;
top = newNode;
}
```

Stack - array based implementation:

```
template<class SIT>
class Stack {
public:
    Stack();
    ~Stack(); // etc.
    bool empty() const;
    void push(const SIT & e);
    SIT pop();
private:
    int capacity;
    int size;
    SIT * items;
};
```

```
template<class SIT>
Stack<SIT>::Stack(){
    capacity = 4;
    size = 0;
    items = new SIT[capacity];
}
```