Announcements

- MP3 due on 02/~~27~~ 26 @11:59pm, E/C due on 02/~~20~~ 19 @11:59p

- Exam 2: 3/2-3/4. 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 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

A pointer in a function parameter list

```
#include <iostream>
using namespace std;

template <class LIT>
struct listNode {
   LIT data;
   listNode *next;
   listNode(LIT d):data(d), next(NULL){};
};

template <class LIT>
void makeNULL(listNode<LIT> * ptr) {


}
```

```
int main() {
   listNode<int> *p = new listNode<int>(5);
   delete p;
   cout << p->data << endl;
   makeNULL(p);
   cout << p->data << endl;
   return 0;
}
```

Problem?

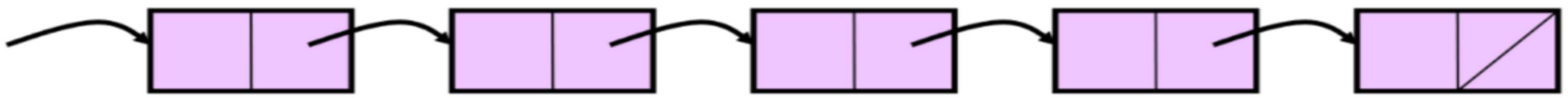
List Functions to Implement:

```
#include <iostream>
using namespace std;

template <class LIT>
struct listNode {
  LIT data;
  listNode *next;
  listNode(LIT d):data(d), next(NULL){};
};

// makeList implementation
// printList implementation
// printReverse implementation
```

```
int main() {
  listNode<int> *h = NULL;

  makeList(h, 3);
  printList(h);
  printReverse(h);

  return 0;
}
```

Problem?

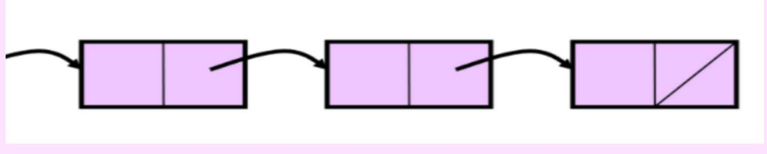
Recursive *makeList* function



```
template <class LIT>
void makeList(listNode<LIT> * curr, int size) {
  if

  }
  return;
}
```

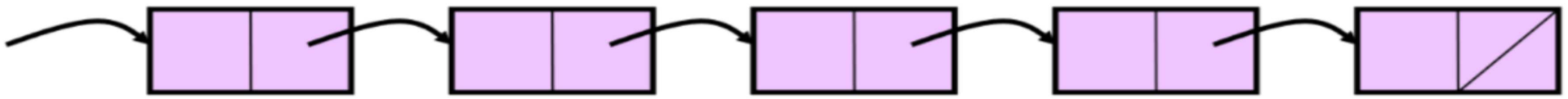
Problem?

```
int main() {
  listNode<int> *h = NULL;

  makeList(h, 3);

  cout << h->data << " " << h->next->data <<
    " " << h->next->next->data << endl;

  return 0;
}
```



Run Time:

Recursive *printList* function



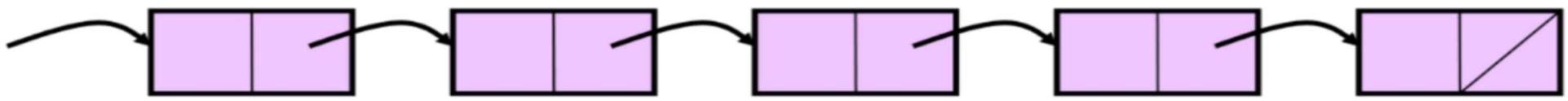
```
template <class LIT>
void printList(listNode<LIT> * curr) {
  if

  }
  else

  return;
}
```

```
int main() {
  listNode<int> *h = NULL;

  makeList(h, 3);
  printList(h);

  return 0;
}
```

Problem?

Run Time:

Recursive *printReverse* function



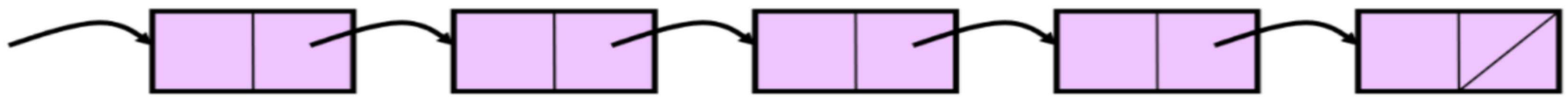
```
template <class LIT>
void printReverse(listNode<LIT> * curr) {
   if

   }
}
```

```
int main() {
   listNode<int> *h = NULL;

   makeList(h, 3);
   printList(h);
   printReverse(h);

   return 0;
}
```

Problem?

Run Time:

Recursive *deleteList* function



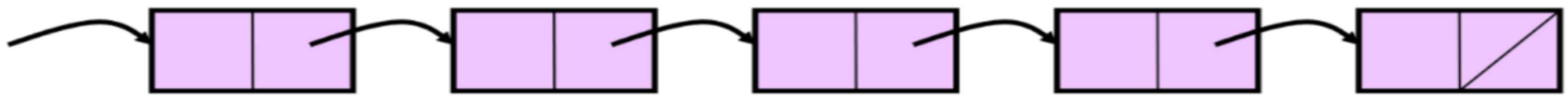
```
template <class LIT>
void deleteList(listNode<LIT> * curr) {
  if

  }
  return;
}
```

Problem?

```
int main() {
  listNode<int> *h = NULL;

  makeList(h, 3);
  printList(h);
  printReverse(h);

  deleteList(h);
  cout << h->data << endl;

  return 0;
}
```

Run Time:

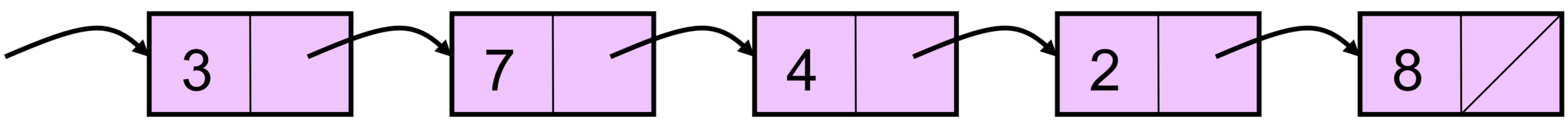
insertAtFront function



```
void insertAtFront(listNode *  curr, LIT e) {

}
```

Running time?

Find kth position (we'll need this later)

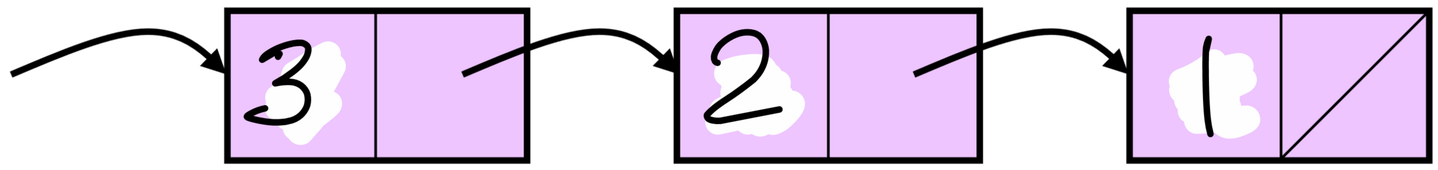



```
//returns pointer to node k steps forward from *curr
listNode * findKth(listNode *  curr, int k) {


}
```

Analysis:

Find kth in array:

Insert new node in kth position:




```
void List<LIT>::insert(int loc, LIT e) {

}
```

Analysis:

insert new kth in array: