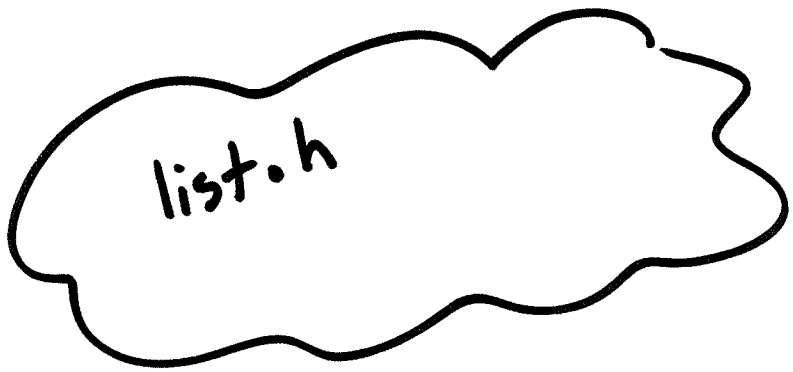
← Prototypes OR #includes

```
class video
{   public:
        video ();
        ~video ();
        int input ();
        int set (video &);
        int display ();
            }

    Private:
        char * title;
            :

};
struct node
{
    video movie;
    node * next;
};
```

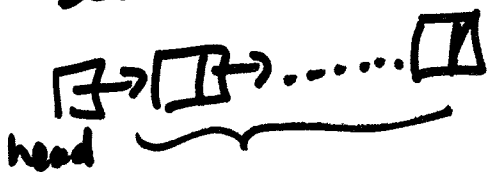
Example #1

```
class list
{
    public:
        list();
        ~list();
        int add (video &);
        int input();
        int display ();
            :

    private:
        node * head;
};
```



```
video::video()
{
    title = NULL;
        :

}
video::~video()
{
    delete [] title;
    title = NULL;
}
int video::set(video & in_v)
{
    title = new char [strlen(in_v.title)+1];
    strcpy(title, in_v.title);
        :
    return 1;
}
```

```
list::list()
{
        head = NULL;
}
list::~list()
{
```




```
    node * temp;
    while (head)
    {       temp = head;
            head = head->next;
            delete temp;

    }
}
```

Implicitly cause video's destructor to be invoked

```
int list::add(video & in_v)
{
    if (head == NULL)
      // if (!head)
    { // empty list
      head = new node;
```

cause video constructor to be implicitly invoked

```
      head->next = NULL;
      head->movie.set(in_v);
```

object of type video

```
    }
    else ....
```

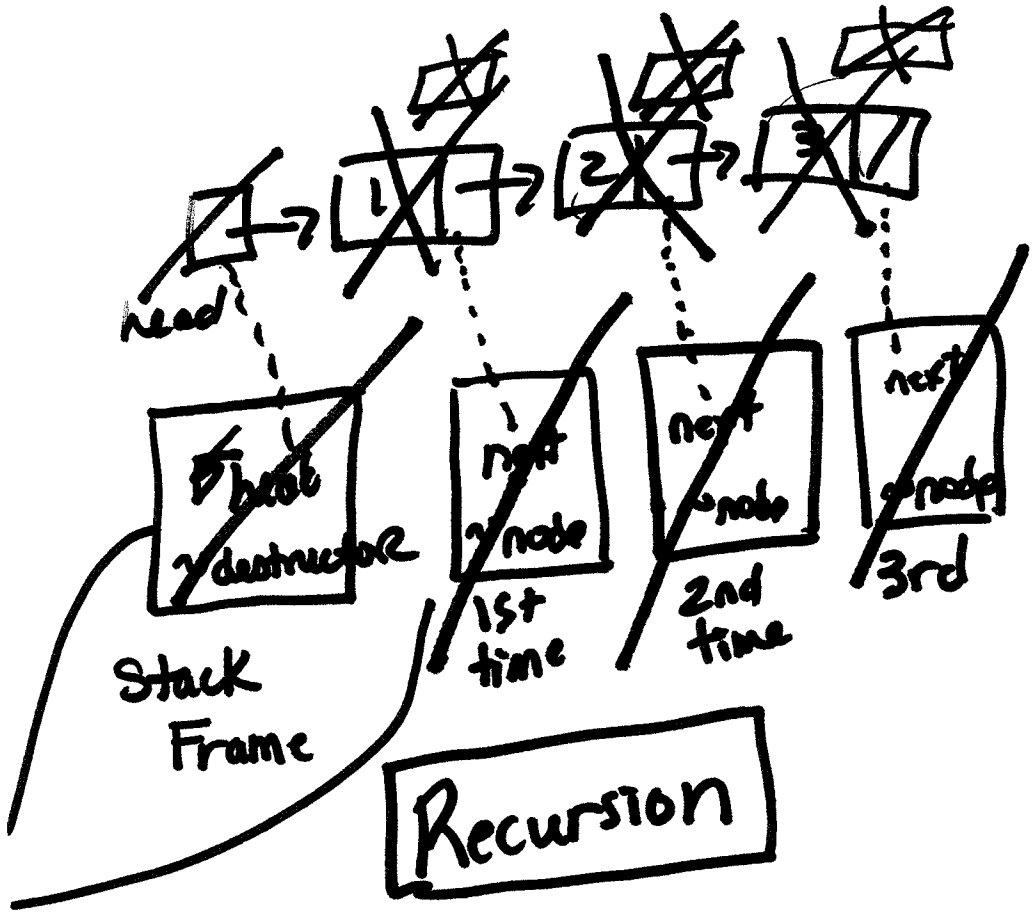
Example #2

```
class video
{

    //same


};
struct node
{   ~node();
      video movie;
      node * next;
};
```

```
class list
{

    //same


};
```

IMPLEMENTATION
- .cpp -

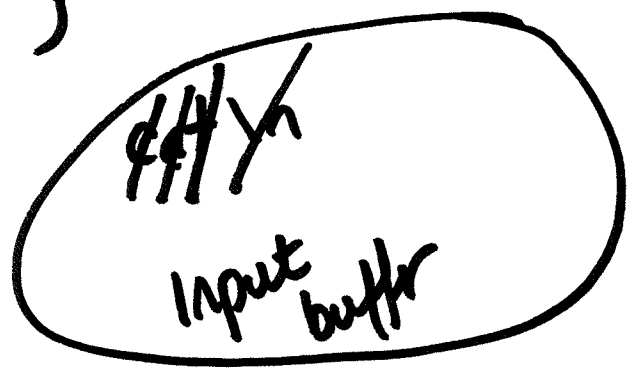
```
node::~node()
{   delete next;
}
list::~list()
{   delete head;
}
```

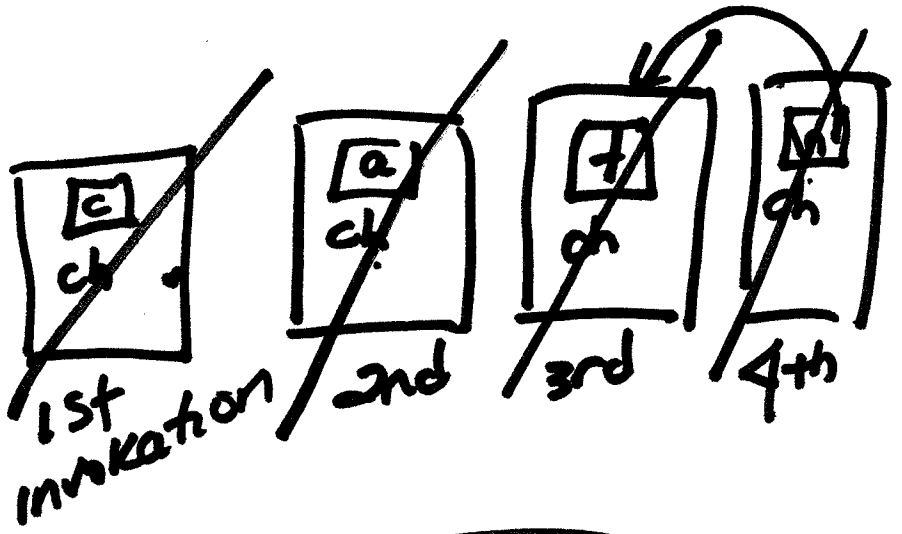


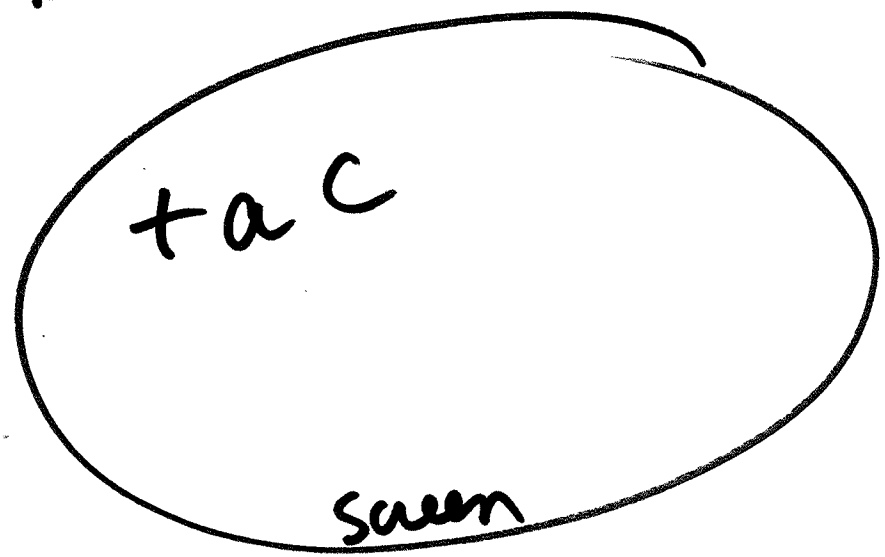
delete implicitly calls destructors to be invoked

Strange

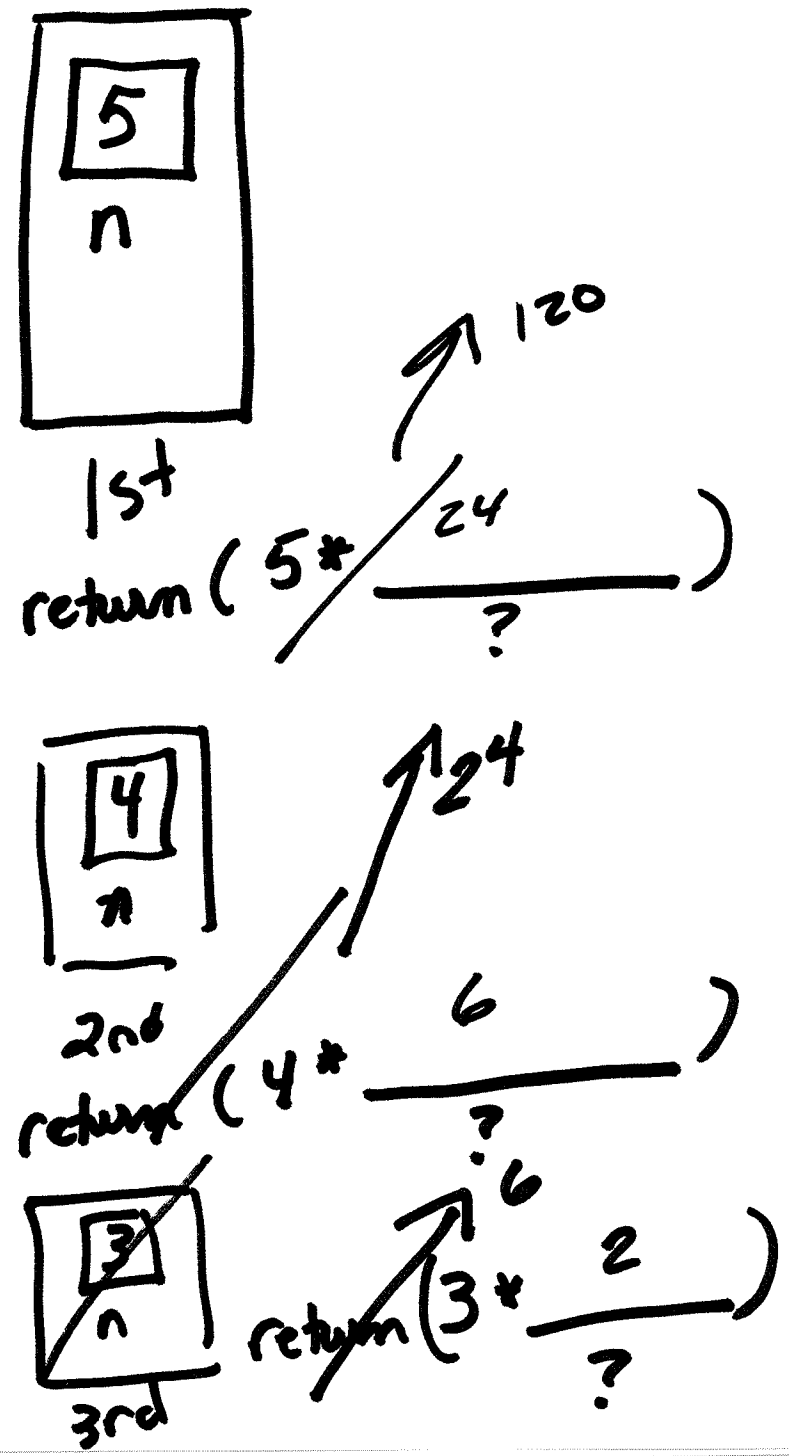
```
void strange (void )
{  char ch;
   cin.get(ch);
   if( ch != '\n' )
   {   strange();
       cout << ch;
   }
}
```

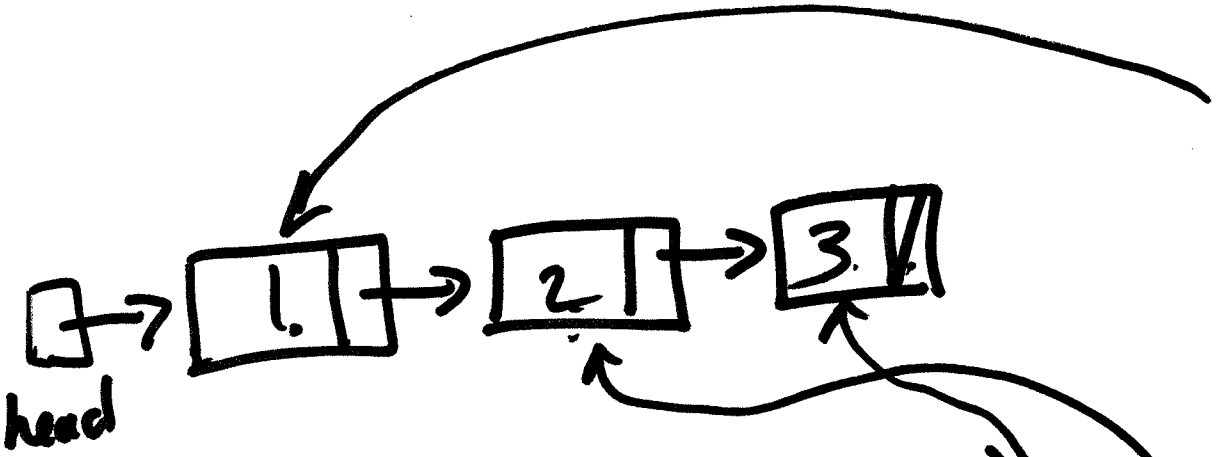






```
int factorial (int n)
{  if (n < 2)
       return 1;

   else
       return (n *
                 factorial (n-1));

}
```

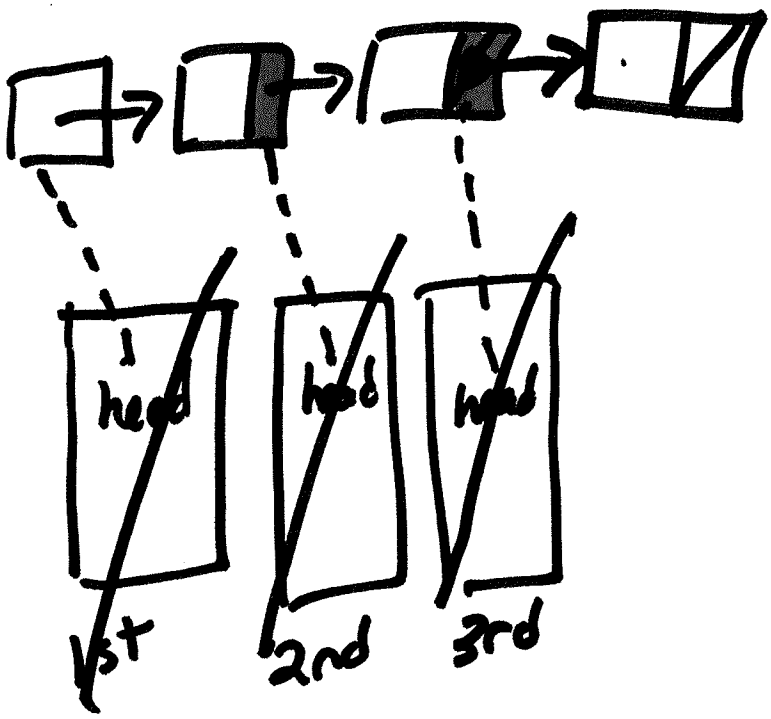





```
void  Something (node * head)
{     if (head)
      {     something (head->next);
            cout << head->data << endl;

      }

}
```

1st
Node
2nd
Node
3rd
Node
4th
Node

```
//Add at end
void add (node * & head)
{
    if (head == NULL)
    {
        head = new node;
        head->next = NULL;
        head->data = ....;
    }
    else
        add (head->next);
}
```


head
head
head
1st
2nd
3rd