Lecture 11:
Self-Reference

"Defining a program or program-structure in terms of itself."

Example: Computing n ! = n (n-1)... 3.2.1

```
#include <stdio.h>
int factorial(int n)
{
 if (n==1)
       return 1;
 else
       return n*factorial(n-1);
}
void main()
{
 printf("%d", factorial(7));
}
```

What does the following declaration do?

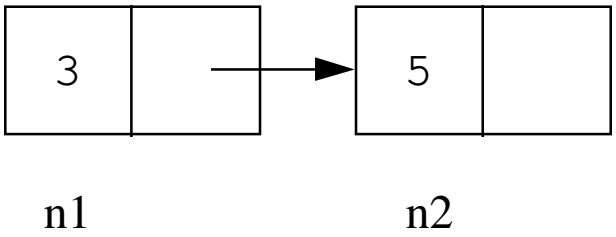
```
#include <stdio.h>
struct node
    {
       int number;
       struct node *next
    };
```

Using self-referential structures

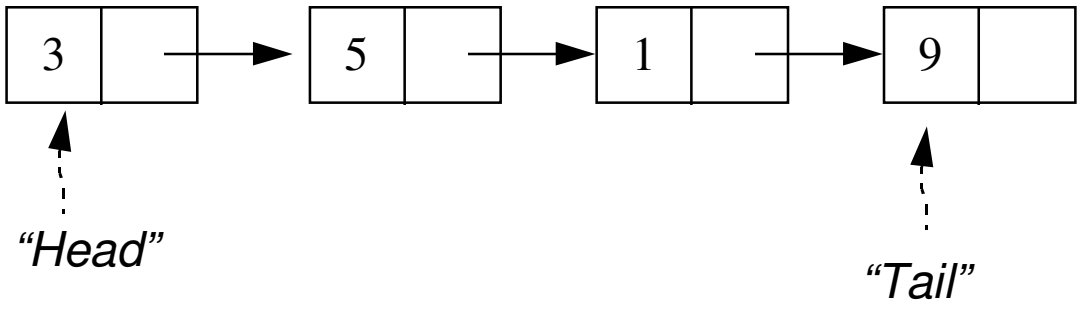
```
#include <stdio.h>
struct node
  {
   int number;
   struct node *next
  };
void main()
{
 struct node n1, n2;
 n1.number =3;
 n2.number =5;
 n1.next = &n2;
}
```

What is

`(*(n1.next)).number?`



Linked Lists




```
#include <stdio.h>
#include <stdlib.h>
struct node
{
        int value;
        struct node *next;
};
struct list
{
        struct node *head;
        struct node *tail;
};
```

Dynamic memory allocation : `malloc()`

`malloc(sizeof(struct node))` : allocates a portion of memory large enough to hold one `node`, and returns the address of that location

Example:

```
#include <stdlib.h>
.
.
.
struct node *newnode;
newnode = malloc(sizeof(struct node));
(*newnode).number =5;
```

```
/* Same Header as before */
void addto(struct list *l, int value)
{
 struct node *newnode;
 newnode = malloc(sizeof(struct node));
 (*newnode).number = value;
 if ((*l).head == NULL)
    (*l).head = (*l).tail =newnode;
 else
    {
      (*(*l).tail).next = newnode;
      (*l).tail = newnode;
    }
}
void main()
{
  struct list mylist;
  mylist.head = NULL;
  addto(&mylist, 3);
  addto(&mylist, 5);
  addto(&mylist, 1);
  addto(&mylist, 9);
}
```

General Philosophy: Write small functions to "maintain" the data structure (`addto, deletefrom, search,..`)

Linked Lists: How to Search Them

```
void search(struct list l, int value)
{
 struct node *current;
 current = l.head;
 while(current != l.tail)
     {
       if ((*current).number == value)
              break;
        current= (*current).next;
     }
  if (*current).number == value)
    printf("Found!\n");
  else
    printf("Not Found!\n");
}
```

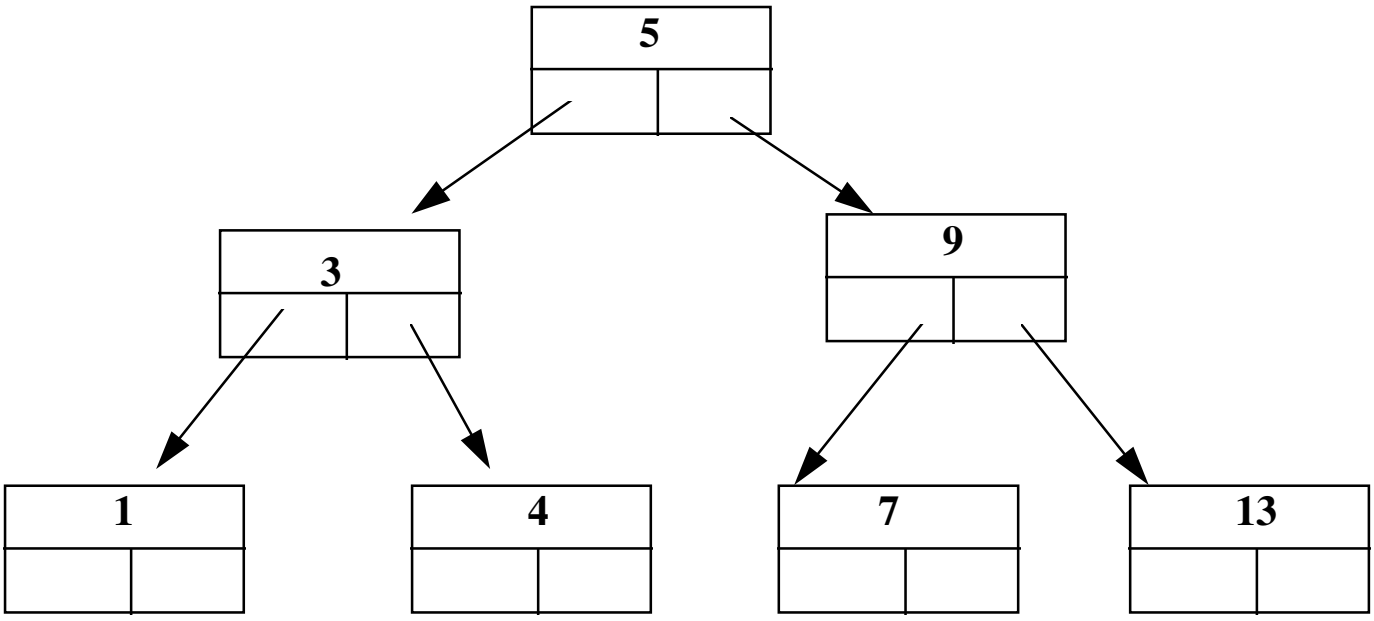
Linked lists: How to reverse them

```
void reverse(struct list *l)
{
 struct node *current;

 if ((*l).head != (*l).tail)
  {
   current = (*l).head;
   (*l).head = (*(*l).head).next;
   reverse(l);
   (*(*l).tail).next =current;
   (*l).tail = current;
  }
}
void main()
{
  struct list mylist;
  /* create a list mylist as before */
  reverse(&mylist);
}
```

Trees

```
struct tree
  {
   int value;
   struct tree *left;
   struct tree *right;
  }
```



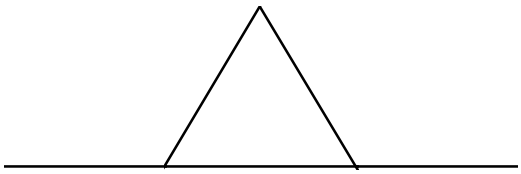
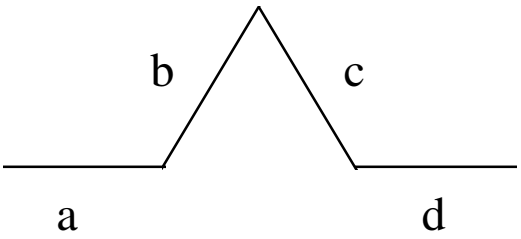
Self-Reference in Math: Koch "Snowflake"

(a "fractal"; defined using self-reference)

Koch(m) :

1. Take a line segment of length m

2. Put an equilateral triangle in the middle whose each side is m/3.



3. Erase the triangle's base and put the curve Koch(m/3) on segments a, b, c, d.